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

Linux From Scratch

Version 3.2−rc2

Gerard Beekmans

This book describes the process of creating a Linux system from scratch from an already installed Linux distribution, using nothing but the sources of the software that we use.

Dedication

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

Table of Contents

Other

Preface

Foreword

Having used a number of different Linux distributions, I was never fully satisfied with any of them. I didn't like the way the bootscripts were arranged, I didn't like the way certain programs were configured by default, and more of those things. I came to realize that if I wanted to be fully satisfied with a Linux system, I would have to build my own system from scratch, ideally using only the source code. Not using pre−compiled packages of any kind. No help from some sort of CD−ROM or bootdisk that would install some basic utilities. I would use my current Linux system and use that one to build my own.

This, at one time, wild idea seemed very difficult and at times almost impossible. After sorting out all kinds of dependency problems, compile problems, etcetera, a custom−built Linux system was created and fully operational. I called this system an LFS system, which stands for Linux From Scratch.

I hope all of you will have a great time working on LFS!

−− Gerard Beekmans gerard@linuxfromscratch.org

Who would want to read this book

There are a lot of reasons why somebody would want to read this book in order to install an LFS system. The question most people raise is "why go through all the hassle of manually installing a Linux system from scratch when you can just download an existing distribution?". That is a valid question which I hope to answer for you.

The most important reason for LFS's existence is teaching people how a Linux system works internally. Building an LFS system teaches you about all that makes Linux tick, how things work together, and depend on each other. And most importantly, how to customize it to your own taste and needs.

One of the key benefits of LFS is that you are in control of your system without having to rely on somebody else's Linux implementation. You are in the driver's seat now and are able to dictate every single thing such as the directory layout and boot script setup. You will also know exactly where, why and how programs are installed.

Another benefit of LFS is that you can create a very compact Linux system. When you install a regular distribution, you end up installing a lot of programs you probably would never use. They're just sitting there taking up (precious) disk space. It's not hard to get an LFS system installed under 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 installed a system that was just enough to run the Apache web server; total disk space usage was approximately 8 MB. With further stripping, that can be brought down to 5 MB or less. Try that with a regular distribution.

If we were to compare a Linux distribution with a hamburger you buy at a supermarket or fast−food restaurant, you would end up eating it without knowing precisely what it is you are eating, whereas LFS gives you the ingredients to make a hamburger. This allows you to carefully inspect it, remove unwanted ingredients, and at the same time allow you to add ingredients to enhance the flavour of your hamburger. When you are satisfied with the ingredients, you go on to the next part of putting it together. You now have the chance to make it just the way you like it: broil it, bake it, deep−fry it, barbeque it, or eat it raw.

Another analogy that we can use is that of comparing LFS with a finished house. LFS will give you the skeleton of a house, but it's up to you to install plumbing, electrical outlets, kitchen, bathtub, wallpaper, etc.

Another advantage of a custom built Linux system is added security. You will compile the entire system from source, thus allowing you to audit everything, if you wish to do so, and apply all the security patches you want or need to apply. You don't have to wait for somebody else to provide a new binary package that fixes a security hole. Besides, you have no guarantee that the new package 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

People who don't want to build an entire Linux system from scratch probably don't want to read this book. If you, however, want to learn more about what happens behind the scenes, in particular what happens between turning on the computer and seeing the command prompt, you may want to read the "From−PowerUp−To−Bash−Prompt−HOWTO". This HOWTO builds a bare system, in a way similar to the one this book uses, but it focuses more on just installing a bootable system instead of a complete system.

To decide whether to read this book or the From−PowerUp−To−Bash−Prompt−HOWTO, ask yourself this question: "Is my main objective to get a working Linux system that I'm going to build myself and, along the way learn what every component of a system is for? Or is just the learning part my main objective?" If you want to build and learn, read this book. If you just want to learn the basics, then the From−PowerUp−To−Bash−Prompt−HOWTO is probably better material to read.

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

Organization

This book is divided into the following parts. Although most of the appendices is copied into part II (which enlarges the book somewhat), we believe it's the easiest way to read it like this. It simply saves you from having to click to an Appendix, then back to where you were in part II. That's a real chore especially if you're reading the TXT version of this book.

Part I – Introduction

Part One gives general information about this book (versions, where to get it, changelog, mailing lists, and how to get in touch with us). It also explains a few important aspects you really want and need to read before starting to build an LFS system.

Part II – Installation of the LFS system

Part Two guides you through the installation of the LFS system which will be the foundation for the rest of the system. Whatever you choose to do with your brand new LFS system, it will be built on the foundation that's installed in this part.

Part III – Appendixes

Part Three contains various Appendices.

I. Part I – Introduction

Table of Contents

Chapter 1. Introduction

Acknowledgments

We would like to 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.
- 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.
- Jesse Tie−Ten−Quee <highos@linuxfromscratch.org> for running the www.ca.linuxfromscratch.org mirror.
- 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 an already installed Linux distribution such as Debian, SuSe, Slackware, Mandrake, RedHat, etc. There is no need to have any kind of bootdisk. We will use an existing Linux system as the base (since we need a compiler, linker, text editor, and other tools).

After you have downloaded the necessary packages that make up an LFS system you will create a new Linux native partition onto which the LFS system will be installed.

The next step, chapter 5, will be the installation of a number of packages that are statically linked and installed on the LFS partition. These packages form a basic development suite which will be used to install the actual system, and are also needed to resolve circular dependencies. Examples of circular dependencies are: you need a compiler to install a compiler. You need a shell in order to install a shell. And so on.

Chapter 6 installs the actual base system. We use the chroot program to start a new shell whose root directory will be set to the LFS partition. This, in essence, is the same as rebooting and having the kernel mount the LFS partition as the root partition. The reason that we don't actually reboot, but instead chroot, is that this way you can still use your host system. While software is being installed you can simply switch to a different VC (Virtual Console) or X desktop and continue using your computer as you normally would.

When all the software is installed, chapter 7 will setup the boot scripts. Chapter 8 will setup the Linux boot loader and in chapter 9 there are some pointers what you can do after you finish the book. Then you can finally reboot your system into your new LFS system, and start to really use it.

This is the process in a nutshell. Detailed information on the steps you are taking are provided in the chapters as you go through them. If something isn't completely clear yet, don't worry. It will become very clear shortly.

Please read chapter 2 carefully as it explains a few important things you need to be aware of before you work your way through chapters 5 and above.

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 referred to.

`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 but mainly to emphasize important points or to give examples as to 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 3.2−rc2 dated February 27th, 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 February 15th, 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 [100 Mbit] – http://www.linuxfromscratch.org/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
- Vienna Univ. of Technology, Austria [16 Mbit] – http://www.at.linuxfromscratch.org/lfs/intro.shtml
- Oslo, Norway [100 Mbit] – http://www.no.linuxfromscratch.org/lfs/intro.shtml
- Teeside, United Kingdom [256 Kbit] – http://www.linuxfromscratch.co.uk/lfs/intro.shtml
- Amsterdam, The Netherlands [100 Mbit] – http://www.nl.linuxfromscratch.org/lfs/intro.shtml

Australia

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

FTP Mirrors

North America

- Fremont, California, USA [FTP] [100 Mbit] – ftp://ftp.linuxfromscratch.org
- Fremont, California, USA [HTTP] [100 Mbit] – http://ftp.linuxfromscratch.org

Europe

- Vienna Univ. of Tech., Austria [FTP] [16 Mbit] – ftp://ftp.at.linuxfromscratch.org/pub/lfs
- Vienna Univ. of Tech., Austria [HTTP] [16 Mbit] – http://ftp.at.linuxfromscratch.org/pub/lfs
- Oslo, Norway [FTP] [100 Mbit] – ftp://ftp.no.linuxfromscratch.org/mirrors/lfs/
- Univ. of Twente, The Netherlands [HTTP] [100 Mbit] – http://ftp.nl.linuxfromscratch.org/linux/lfs
- Univ. of Twente, The Netherlands [FTP] [100 Mbit] – ftp:/ftp.nl.linuxfromscratch.org/pub/linux/lfs

Australia

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

Changelog

3.2−rc2 − February 27th, 2002

- Updated to:
  - lfs−bootscripts−1.6
- 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 setup 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
- 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.

alfs−discuss

The alfs−discuss list discusses the development of ALFS, which stands for Automated Linux From Scratch. The goal of this project is to develop an installation tool that can install an LFS system automatically. Its main goal is to speed up compilation by taking away the need to manually enter the commands to configure, compile, and install packages.

blfs−dev

The blfs−dev mailing list discusses matters related to the BLFS−BOOK (Beyond LFS). If problems with the book come up, a bug or two need to be reported, or suggestions to improve the book (such as suggestions as to installation instructions to add) are to be made, this mailing list is the right one.

Requests for help with programs beyond the base LFS setup (not just those in BLFS) should go to blfs−support.

blfs−book

The blfs−book list is used by the BLFS−BOOK editors to co−ordinate blfs−book's maintenance, like XML 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 deals with support requests for any software not installed in the LFS book. The list is not just for help with software explicitly mentioned in the BLFS book, any software beyond that 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 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 problems building an LFS system, you should check out http://www.linuxfromscratch.org/faq/ to see if your question is already answered in the 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.

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

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 like **`./configure --prefix=$LFS`** you can type that 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.

How to download the software

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

A convention you could use is having a $LFS/usr/src/sources directory. Under sources, you can create the directory 0−9 and the directories a through z. A package like sysvinit−2.84.tar.bz2 is stored under $LFS/usr/src/sources/s/. A package like bash−2.05a.tar.bz2 is stored under $LFS/usr/src/sources/b/, and so forth.

The next chapter contains the list of all the packages that need to be downloaded, but the partition that is going to contain our LFS system isn't created yet. Therefore, you should store the files somewhere else and later on move them to $LFS/usr/src/ when the chapter in which the new partition is prepared has been finished.

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
```

Some tar programs (most of them nowadays but not all of them) are slightly modified to be able to use bzip2 files directly using either the I, the y or the j tar parameter, which works the same as the z tar parameter to handle gzip archives. The above construction works no matter how your host system decided to patch bzip2.

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 first.

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 for use 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'ed 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.

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 will use the kernel tree so the source tree won't be in your way.

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 have a problem while using this book, you'll find that most of the people on Internet Relay Chat (IRC) and the mailing lists will be willing to help you. You can find a list of the LFS mailing lists in Chapter 1 − Mailing lists and archives. To assist us in helping though, you should make sure that you have as much relevant information as you can available. This will assist in diagnosing and solving your problem. This part of the book will guide you as to which sort of information will be useful.

Basic Information

First of all we need a brief explanation of the problem. Essential things to include are:

- The version of the book you are using, which is 3.2−rc2
- Which package or section you are having problems with
- What the exact error message or symptom you are receiving is
- If 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 all about choice. It'll just help us to see the possible other 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 contains possible errors encountered during configure which aren't always 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 when make is 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 quoted above is what should be included to be helpful, because it includes the command that was executed and the command's error message(s).

Download the LFS Commands

LFS Commands is a tarball containing files which list the installation commands for the packages installed in this book.

These files can be used to quickly find out which commands have been changed between the different LFS versions. Download the lfs−commands tarball for this book version and the previous book version and run a diff on the files. That way it is possible to see which packages have updated installation instructions, so any scripts you may have can be modified, or you can reinstall a package if you think that necessary.

A side effect is that these files can be used to dump to a shell and install the packages, though some files need to be modified (where certain settings can't be guessed and depend on user preference or system hardware). Keep in mind, please, that these files are not thoroughly checked for correctness. There may be bugs in the files (since they are manually created at the moment) so do check them and don't blindly trust them.

If you decide to use the commands to automatically install a package and it doesn't work, try reading the book's instructions instead before you ask for help on the mailinglist.

The lfscommands can be downloaded from http://ftp.linuxfromscratch.org/lfs−commands/ or ftp://ftp.linuxfromscratch.org/lfs−commands/ .

II. Part II – Installing the LFS system

Table of Contents

Chapter 3. Packages that need to be downloaded

Introduction

Below is a list of all the packages that are needed to download for building the basic system. The version numbers printed correspond to versions of the software that is known to work and which this book is based on. If you experience problems which you can't solve yourself, then please download the version that is assumed in this book (in case you downloaded a newer version).

All the URL's below are to the ftp.linuxfromscratch.org server. We have a couple of FTP mirrors available from which you can download the files a well. The addresses of the mirror sites can be found in Chapter 1 – Book Version.

We have provided a list of official download sites of the packages below in Appendix A. The LFS FTP archive only contains the versions of packages that are recommended for use in this book. You can check the official sites in Appendix A to determine whether a newer package is available. If you do download a newer package, we would appreciate hearing whether you were able to install the package using this book's instructions or not.

Please note that all files downloaded from the LFS FTP archive are files compressed with bzip2 instead of gz. If you don't know how to handle bz2 files, check out Chapter 2 – How to install the software.

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 − 81,170 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/lfs−packages−3.2−rc2.tar
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/lfs−packages−3.2−rc2.tar
Or download the following packages individually: Bash (2.05a) − 1,400 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bash−2.05a.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bash−2.05a.tar.bz2 Binutils (2.11.2) − 7,641 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/binutils−2.11.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/binutils−2.11.2.tar.bz2 Bzip2 (1.0.1) − 410 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bzip2−1.0.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bzip2−1.0.1.tar.bz2 Diff Utils (2.7) − 247 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/diffutils−2.7.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/diffutils−2.7.tar.bz2 File Utils (4.1) − 1217 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/fileutils−4.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/fileutils−4.1.tar.bz2 GCC (2.95.3) − 9,618 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gcc−2.95.3.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gcc−2.95.3.tar.bz2 GCC Patch (2.95.3−2) − 8 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gcc−2.95.3−2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gcc−2.95.3−2.patch.bz2
Linux Kernel (2.4.17) − 23,282 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/linux−2.4.17.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/linux−2.4.17.tar.bz2 Grep (2.4.2) − 382 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/grep−2.4.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/grep−2.4.2.tar.bz2 Gzip (1.2.4a) − 178 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gzip−1.2.4a.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gzip−1.2.4a.tar.bz2 Gzip Patch (1.2.4a) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gzip−1.2.4a.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gzip−1.2.4a.patch.bz2 Make (3.79.1) − 794 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/make−3.79.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/make−3.79.1.tar.bz2 Sed (3.02) − 221 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sed−3.02.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sed−3.02.tar.bz2 Sh−utils (2.0) − 824 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sh−utils−2.0.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sh−utils−2.0.tar.bz2 Sh−utils Patch (2.0) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sh−utils−2.0.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sh−utils−2.0.patch.bz2 Tar (1.13) − 730 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/tar−1.13.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/tar−1.13.tar.bz2 Tar Patch (1.13) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/tar−1.13.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/tar−1.13.patch.bz2 Text Utils (2.0) − 1,040 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/textutils−2.0.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/textutils−2.0.tar.bz2 Mawk (1.3.3) − 168 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/mawk1.3.3.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/mawk1.3.3.tar.bz2 Texinfo (4.0) − 812 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/texinfo−4.0.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/texinfo−4.0.tar.bz2 Patch (2.5.4) − 149 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/patch−2.5.4.tar.bz2

http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/patch−2.5.4.tar.bz2 MAKEDEV (1.4) − 7 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/MAKEDEV−1.4.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/MAKEDEV−1.4.bz2 Glibc (2.2.5) − 12,114 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/glibc−2.2.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/glibc−2.2.5.tar.bz2
Glibc−linuxthreads (2.2.5) − 164 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/glibc−linuxthreads−2.2.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/glibc−linuxthreads−2.2.5.tar.bz2
Man−pages (1.47) − 534 KB: ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/man−pages−1.47.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/man−pages−1.47.tar.bz2 Ed (0.2) − 158 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ed−0.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ed−0.2.tar.bz2 Find Utils (4.1) − 226 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/findutils−4.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/findutils−4.1.tar.bz2 Find Utils Patch (4.1) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/findutils−4.1.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/findutils−4.1.patch.bz2 Ncurses (5.2) − 1,308 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ncurses−5.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ncurses−5.2.tar.bz2 Vim (6.0) − 2,711 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/vim−6.0.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/vim−6.0.tar.bz2 Bison (1.31) − 510 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bison−1.31.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bison−1.31.tar.bz2 Less (358) − 178 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/less−358.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/less−358.tar.bz2 Groff (1.17.2) − 1,214 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/groff−1.17.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/groff−1.17.2.tar.bz2 Man (1.5j) − 167 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/man−1.5j.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/man−1.5j.tar.bz2 Perl (5.6.1) − 4,750 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/perl−5.6.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/perl−5.6.1.tar.bz2 M4 (1.4) − 249 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/m4−1.4.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/m4−1.4.tar.bz2 Autoconf (2.52) − 618 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/autoconf−2.52.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/autoconf−2.52.tar.bz2 Automake (1.5) − 409 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/automake−1.5.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/automake−1.5.tar.bz2 Flex (2.5.4a) − 278 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/flex−2.5.4a.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/flex−2.5.4a.tar.bz2 File (3.37) − 140 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/file−3.37.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/file−3.37.tar.bz2 Libtool (1.4.2) − 653 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/libtool−1.4.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/libtool−1.4.2.tar.bz2 Bin86 (0.16.0) − 113 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bin86−0.16.0.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/bin86−0.16.0.tar.bz2 Gettext (0.10.40) − 941 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gettext−0.10.40.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/gettext−0.10.40.tar.bz2 Kbd (1.06) − 559 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/kbd−1.06.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/kbd−1.06.tar.bz2 Kbd Patch (1.06−2) − 3 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/kbd−1.06−2.patch.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/kbd−1.06−2.patch.bz2 E2fsprogs (1.25) − 1,029 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/e2fsprogs−1.25.tar.bz2

http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/e2fsprogs−1.25.tar.bz2 Lilo (22.1) − 262 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/lilo−22.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/lilo−22.1.tar.bz2 Modutils (2.4.12) − 209 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/modutils−2.4.12.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/modutils−2.4.12.tar.bz2 Procinfo (18) − 22 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/procinfo−18.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/procinfo−18.tar.bz2 Procps (2.0.7) − 153 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/procps−2.0.7.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/procps−2.0.7.tar.bz2 Psmisc (20.2) − 123 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/psmisc−20.2.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/psmisc−20.2.tar.bz2 Reiserfsprogs (3.x.0j) − 196 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/reiserfsprogs−3.x.0j.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/reiserfsprogs−3.x.0j.tar.bz2
Shadow Password Suite (20001016) − 551 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/shadow−20001016.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/shadow−20001016.tar.bz2 Sysklogd (1.4.1) − 67 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sysklogd−1.4.1.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sysklogd−1.4.1.tar.bz2 Sysvinit (2.84) − 76 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sysvinit−2.84.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/sysvinit−2.84.tar.bz2 Util Linux (2.11n) − 998 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/util−linux−2.11n.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/util−linux−2.11n.tar.bz2 Netkit−base (0.17) − 49 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/netkit−base−0.17.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/netkit−base−0.17.tar.bz2 Net−tools (1.60) − 194 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/net−tools−1.60.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/net−tools−1.60.tar.bz2
LFS−Bootscripts (1.6) − 6 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/lfs−bootscripts−1.6.tar.bz2
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/lfs−bootscripts−1.6.tar.bz2
Total size of all packages: 81,170 KB (79.27 MB)

Chapter 4. Preparing a new partition

Introduction

In this chapter, the partition that is going to host the LFS system is going to be prepared. We will be creating the partition itself, a file system and the directory structure. When this is done, we can move on to the next chapter and start the actual building process.

Creating a new partition

First, let's start with telling you that it is possible to build LFS on only one partition, which is where your original distribution is installed. This is not recommended if it is the first time you try LFS, but 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 to have an empty Linux partition on which we can build our new system. We recommend a partition size of around 1 GB. This gives enough space to store all the tarballs and to compile all packages without worrying about running out of the necessary temporary disk space. But you probably want more space than that if you plan to use the LFS system as your primary Linux system. If that's the case you'd want more space so you can install additional software. If a Linux Native partition is already available, this subsection can be skipped.

The cfdisk program (or another fdisk like program you prefer) is to be started with the appropriate hard disk as the option (like /dev/hda if a new partition is to be created on the primary master IDE disk). It is used to create a Linux Native partition, write the partition table and exit the cfdisk program. Please refer to the documentation that comes with your fdisk program of choice (the man pages are often a good place to start) and read the procedures about how to create a new Linux native partition and how to write the partition table.

The new partition's designation should be remembered. It could be something like 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. It's of course up to you to decide which file system you want to create, but because we have to assume and work with something, we will assume you chose the ext2 file system.

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.

Chapter 5. Preparing the LFS system

Introduction

In the following chapters we will install all the software that belongs to a basic Linux system. After you're done with this and the next chapter, you'll have a fully working Linux system. The remaining chapters deal with creating the boot scripts, making the LFS system bootable and setting up basic networking.

The software in this chapter will be linked statically and will be reinstalled in the next chapter and linked dynamically. The reason for the static version first is that there is a chance that our normal Linux system and the LFS system aren't using the same C Library versions. If the programs in the first part are linked against an older C library version, those programs might not work well on the LFS system. Another reason is to resolve circular dependencies. An example of such a dependency is that you need a compiler to install a compiler, and you're going to need a shell to install a shell and that compiler.

The key to learning what makes Linux tick is to know exactly what packages are used for and why a user or the system needs them. Descriptions of the package content are provided after the Installation subsection of each package and in Appendix A as well.

During the installation of various packages, you will more than likely see all kinds of compiler warnings scrolling by on the screen. These are normal and can be safely ignored. They are just that, warnings (mostly about improper use of the C or C++ syntax, but not illegal use. It's just that, often, C standards changed and packages still use the old standard which is not a problem).

Before we start, make sure the LFS environment variable is setup 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 from can often be around 2.5MB), 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 2.5MB attached to it! Instead of making the library an actual part of the program, or statically linked, the library is kept 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 1kb file and a 2.5MB file, but we still haven't saved any space (except maybe RAM until the library is needed). The REAL advantage to 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, both 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.

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 it's time to change the permissions on your LFS partitions so user "lfs" will have write access to it. Run the following command as root to change the ownership of the LFS partition to user "lfs":

```
chown -R lfs $LFS
```

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 setup a good environment to work in:

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

LFS=/mnt/lfs
LC_ALL=POSIX
export LFS LC_ALL
EOF
source ~/.bash_profile
```

This profile makes sure the umask is set to 022 so newly created files and directories will have the correct permission. It is advisable to keep this setting throughout your LFS installation. Also, the $LFS and $LC_ALL 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 else 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.

Creating directories

Let's now create the directory tree on the LFS partition based on the FHS standard, which can be found at http://www.pathname.com/fhs/. Issuing the following commands will create a default directory layout:

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

Normally, directories are created with permission mode 755, which isn't desired for all directories. The first change is a mode 0750 for the $LFS/root directory. This is to make sure that not just everybody can enter the /root directory (the same a user would do with /home/username directories). The second change is a mode 1777 for the tmp directories. This way, any user can write data to the /tmp or /var/tmp directory but cannot remove another user's files (the latter is caused by the so−called "sticky bit" − bit 1 of the 1777 bit mask).

```
cd $LFS &&
chmod 0750 root &&
chmod 1777 tmp var/tmp
```

Now that the directories are created, copy the source files that were downloaded in chapter 3 to some subdirectory under $LFS/usr/src (you will need to create the desired directory yourself).

FHS compliance notes

The FHS stipulates that the /usr/local directory should contain the bin, games, include, lib, man, sbin, and share subdirectories. You can alter your /usr/local directory yourself if you want your system to be FHS−compliant.

Also, the standard says that there should exist a /usr/share/games directory, which we don't much like for a base system. But feel free to make your system FHS−compliant if you wish. The FHS isn't precise as to the structure of the /usr/local/share subdirectories, so we took the liberty of creating the directories that we felt needed.

Installing Bash−2.05a

```
Estimated build time:           3 minutes
Estimated required disk space:  20 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:

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

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

```
./configure --enable-static-link --prefix=$LFS/usr \
   --bindir=$LFS/bin --with-curses &&
make &&
make install &&
cd $LFS/bin &&
ln -sf bash sh
```

If the make install phase ends with something along the lines of

```
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.

When we tested it with the latest Debian version, the last two commands were executed because the install process didn't return with a value larger than 0. But you would do good to check if you have the `$LFS/bin/sh` symlink on your LFS partition. If not, run the last two commands manually now.

Command explanations

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

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

--bindir=$LFS/bin: This installs the executable files in $LFS/bin. We do this because we want bash to be in /bin, not in /usr/bin. One reason being: the /usr partition might be on a separate partition which has to be mounted at some point. Before that partition is mounted you need and will want to have bash available (it will be hard to execute the boot scripts without a shell for instance).

--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.

ln -sf bash sh: This command creates the sh symlink that points to bash. Most scripts run themselves via 'sh' (invoked by the #!/bin/sh as the first line in the scripts) which invokes a special bash mode. Bash will then behave (as closely as possible) as the original Bourne shell.

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

Contents of bash−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. Bash is a program that 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.

Dependencies

Bash−2.05a needs the following to be installed:

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 mawk: awk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info textutils: cat, tr, uniq

Installing Binutils−2.11.2

```
Estimated build time:           6 minutes
Estimated required disk space:  96 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, so we recommend you unsetting CFLAGS, CXXFLAGS and other such variables/settings that would change the default optimization that it comes with.

Install Binutils by running the following commands:

```
mkdir ../binutils-build &&
cd ../binutils-build &&
../binutils-2.11.2/configure --prefix=$LFS/usr --disable-nls &&
make LDFLAGS=-all-static tooldir=$LFS/usr &&
make tooldir=$LFS/usr install
```

Command explanations

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

`--disable-nls:` This option disabled 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.

`tooldir=$LFS/usr:` Normally, the tooldir (the directory where the executables from binutils end up in) 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 $LFS/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).

Contents of binutils−2.11.2

Program Files

addr2line, ar, as, c++filt, 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.

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.

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 a member of an archive 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 objfile 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, libiberty.a and libopcodes.a

Descriptions

libbfd

libbfd is the Binary File Descriptor library.

libiberty

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

libopcodes

No description is currently available.

Dependencies

Binutils−2.11.2 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, true, uname texinfo: install−info, makeinfo
textutils: cat, sort, tr, uniq

Installing Bzip2−1.0.1

```
Estimated build time:           1 minute
Estimated required disk space:  3 MB
```

Installation of Bzip2

Install Bzip2 by running the following commands:

```
make CC="gcc −static" &&
make PREFIX=$LFS/usr install &&
cd $LFS/usr/bin &&
mv bzcat bunzip2 bzip2 bzip2recover $LFS/bin
```

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 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 bzip2−1.0.1

Program Files

bunzip2 (link to bzip2), bzcat (link to bzip2), bzip2 and bzip2recover

Descriptions

bunzip2

Bunzip2 decompresses files that are compressed with bzip2.

bzcat

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

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.

Library Files

libbz2.[a,so]

libbz2

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

Dependencies

Bzip2−1.0.1 needs the following to be installed:

bash: sh binutils: ar, as, ld, ranlib fileutils: cp, ln, rm gcc: cc1, collect2, cpp0, gcc make: make

Installing Diffutils−2.7

```
Estimated build time:           1 minute
Estimated required disk space:  4 MB
```

Installation of Diffutils

When installing Diffutils using glibc−2.1.x on your base system, it may be necessary to use a fix to prevent a variable name conflict. The following commands can be used in this case. Note that these commands can also be used for other glibc versions so if you aren't sure, then use the first version.

```
export CPPFLAGS=-Dre_max_failures=re_max_failures2 &&
./configure --prefix=$LFS/usr &&
unset CPPFLAGS &&
make LDFLAGS=-static PR_PROGRAM=/usr/bin/pr &&
make install
```

If you are using a newer glibc version (2.2.x), you can use the following commands to install Diffutils:

```
./configure --prefix=$LFS/usr &&
make LDFLAGS=-static PR_PROGRAM=/usr/bin/pr &&
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.

`PR_PROGRAM=/usr/bin/pr:` When 'diff' is called with the '−l' flag, it tries to find 'pr' at the location specified with the PR_PROGRAM variable. By default, this variable points to '/bin/pr', but since we install 'pr' in /usr/bin, we need to change this.

Contents of diffutils−2.7

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.

Dependencies

Diffutils−2.7 needs the following to be installed:

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:           3 minutes
Estimated required disk space:  25 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 or higher installed. It also 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 under this category, try the following fix.

Note that in some cases using these sed commands will result in problems 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 the time.

To fix this package to compile properly on AMD/Glibc−2.2.3 machines, run the following commands. 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.

```
cp lib/Makefile.in lib/Makefile.in.backup &&
sed -e 's/\(.*\)\(fopen-safer\.c \)\\/\1\2atexit.c \\/' \
   -e 's/\(.*\)\(idcache\$U\.\$.*\)\\/\1\2atexit$U.$(OBJEXT) \\/' \
   lib/Makefile.in.backup > lib/Makefile.in
```

Install fileutils by running the following commands:

```
./configure --disable-nls \
   --prefix=$LFS/usr --bindir=$LFS/bin &&
make LDFLAGS=-static &&
make install &&
cd $LFS/usr/bin &&
ln -sf ../../bin/install
```

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

Command explanations

`cp lib/Makefile.in lib/Makefile.in.backup` **:** We run this command in order to keep a backup of the file we are about to change.

`cp lib/Makefile.in lib/Makefile.in.backup && sed -e 's/\(.*\)\(fopen-safer\.c \)\\/\1\2atexit.c \\/' \ -e 's/\(.*\)\(idcache\$U\.\$.*\)\\/\1\2atexit$U.$(OBJEXT) \\/' \ lib/Makefile.in.backup > lib/Makefile.in`: 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−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 argument and for each subdirectory of directory arguments.

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.

Dependencies

Fileutils−4.1 needs the following to be installed:

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 GCC−2.95.3

```
Estimated build time:           22 minutes
Estimated required disk space:  168 MB
```

Installation of GCC

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, so we recommend you unsetting CFLAGS, CXXFLAGS and other such variables/settings that would change the default optimization that it comes with.

Install GCC by running the following commands:

```
patch -Np1 -i ../gcc-2.95.3-2.patch &&
mkdir ../gcc-build &&
cd ../gcc-build &&
../gcc-2.95.3/configure --prefix=/usr --enable-languages=c,c++ \
   --disable-nls --disable-shared --enable-threads=posix &&
make BOOT_LDFLAGS=-static bootstrap &&
make prefix=$LFS/usr install &&
cd $LFS/lib &&
ln -sf ../usr/bin/cpp &&
cd $LFS/usr/lib &&
ln -sf ../bin/cpp &&
cd $LFS/usr/bin &&
ln -sf gcc cc
```

Command explanations

patch -Np1 -i ../gcc-2.95.3-2.patch: This new patch deals with incorrect handling of weak symbols, the over−optimization of calls to those weak symbols, an atexit issue and the __dso_handle symbol required for atexit's proper function.

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

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

--enable-languages=c,c++: This only builds the C and C++ compilers and not the other available compilers as they are, on the average, not often used. If those other compilers are needed, the −−enable−languages parameter can be omitted.

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

ln -sf ../usr/bin/cpp: This creates the $LFS/lib/cpp symlink. Some packages explicitly try to find cpp in /lib.

ln -sf ../bin/cpp: This creates the $LFS/usr/lib/cpp symlink as there are packages that expect cpp to be in /usr/lib.

Contents of gcc−2.95.3

Program Files

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

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

c++filt is used to demangle C++ symbols.

collect2

No description is currently available.

cpp, cpp0

cpp pre−processes a source file, such as including the contents of header files into the source file. It's a good idea to not do this manually to save a lot of time. Someone just inserts a line like #include <filename>. The preprocessor inserts the contents of that file into the source file. That's one of the things a preprocessor does.

gcov

No description is currently available.

protoize

No description is currently available.

unprotoize

No description is currently available.

Library Files

libgcc.a, libiberty.a, libstdc++.[a,so]

libgcc

libgcc.a is a run−time support file for gcc.Most of the time, on most machines, libgcc.a is not actually necessary.

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.

Dependencies

GCC−2.95.3 needs the following to be installed:

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−2.4.2

```
Estimated build time:           1 minute
Estimated required disk space:  4 MB
```

Installation of Grep

When installing Grep using glibc−2.1.x on your base system, it may be necessary to use a fix to prevent a variable name conflict. The following commands can be used in this case. Note that these commands can also be used for other glibc versions so if you aren't sure, then use the first version.

```
export CPPFLAGS=-Dre_max_failures=re_max_failures2 &&
./configure --prefix=$LFS/usr --disable-nls --bindir=$LFS/bin &&
unset CPPFLAGS &&
make LDFLAGS=-static &&
make install
```

If you are using a newer glibc version (2.2.x), you can use the following commands to install Grep:

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

Contents of grep−2.4.2

Program Files

egrep, fgrep 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.

Dependencies

Grep−2.4.2 needs the following to be installed:

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

mawk: mawk 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:           1 minute
Estimated required disk space:  2 MB
```

Installation of Gzip

Before Gzip is installed, the patch file may need to be applied. This patch file is necessary to avoid a conflict of variable names with Glibc−2.0 systems when compiling and linking statically and so is only required if your base system runs Glibc−2.0. 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.

Apply the patch by running the following command:

```
patch −Np1 −i ../gzip−1.2.4a.patch
```

Install Gzip by running the following commands:

```
./configure --prefix=$LFS/usr &&
make LDFLAGS=-static &&
make install &&
cp $LFS/usr/bin/gunzip $LFS/usr/bin/gzip $LFS/bin &&
rm $LFS/usr/bin/gunzip $LFS/usr/bin/gzip
```

Command explanations

`cp $LFS/usr/bin/gunzip $LFS/usr/bin/gzip $LFS/bin && rm $LFS/usr/bin/gunzip $LFS/usr/bin/gzip`: The reason we don't simply use "mv" to move the files to the new location is because gunzip is a hardlink to gzip. On older distributions you can't move a hardlink to another partition (and it's very possible that $LFS and $LFS/usr are separate partitions). With more recent distributions this isn't a problem. If you run mv to move hardlinks across partitions it'll just do a regular "cp" and discard the hardlink. But, we can't assume that every host distribution has a new enough kernel and fileutils that works this way.

Contents of gzip−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 either a list of files on the command line or its standard input and writes the uncompressed data on standard output

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.

Dependencies

Gzip−1.2.4a needs the following to be installed:

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 Linux Kernel−2.4.17

```
Estimated build time:           3 minutes
Estimated required disk space:  132 MB
```

Installation of the Linux Kernel

We won't be compiling a new kernel image yet. We'll do that after we have finished the installation of the basic system software in this chapter. But because certain software needs the kernel header files, we're going to unpack the kernel archive now and set it up so that we can compile the packages that need the kernel.

The kernel configuration file is created by running the following command:

```
make mrproper &&
yes "" | make config &&
make dep &&
mkdir $LFS/usr/include/asm &&
cp include/asm/* $LFS/usr/include/asm &&
cp −R include/linux $LFS/usr/include
```

Command explanations

`make mrproper:` This will ensure that the kernel tree is absolutely clean. We do this because the kernel team recommend that this is done prior to *each* kernel compilation, and that we shouldn't rely on the source tree being automatically clean after untarring.

`yes "" | make config:` This runs make config and answers with the default answer to every question the config script asks the user (it does this by simply doing the equivalent of hitting the Enter key, thus accepting the default Y and N answers to the questions). We're not configuring the real kernel here, we just need to have some sort of configure file created so that we can run make dep next that will create a few header files in `include/linux`, like version.h, among others, that we will need to compile Glibc and other packages later in chroot.

`make dep:` make dep checks dependencies and sets up the dependencies file. We don't really care about the dependency checks, but what we do care about is that make dep creates those aforementioned files in `include/linux` we will be needing later on.

`mkdir $LFS/usr/include/asm` and **`cp include/asm/* $LFS/usr/include/asm`**: This copies the platform−specific assembler kernel header files to `$LFS/usr/include/asm`.

`cp −R include/linux $LFS/usr/include`: This command copies the cross−platform kernel header files to `$LFS/usr/include`.

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

In the past, it was common practice for people to symlink the /usr/include/linux and asm directories to /usr/src/linux/include/linux and asm respectively. This is a *bad* idea as this 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 relevant part here is where he states that the headers should be the ones which *glibc was compiled against*. These are the headers which should remain accessible and so by copying them, we ensure that we follow these guidelines. Also note that as long as you don't have those symlinks, it is perfectly fine to have the kernel sources in `/usr/src/linux`.

Contents of kernel−2.4.17

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 such as 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 so should *not* be replaced when upgrading the kernel. They are essential for compiling many programs.

Dependencies

Linux−2.4.17 needs the following to be installed:

bash: sh binutils: ar, as, ld, nm, objcopy fileutils: cp, ln, mkdir, mv, rm findutils: find, xargs
gcc: cc1, collect2, cpp0, gcc grep: grep gzip: gzip make: make mawk: 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, touch

Installing Make−3.79.1

```
Estimated build time:           1 minute
Estimated required disk space:  6 MB
```

Installation of Make

Install Make by running the following commands:

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

Contents of make−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.

Dependencies

Make−3.79.1 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info, makeinfo
textutils: cat, tr

Installing Mawk−1.3.3

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Mawk

Install Mawk by running the following commands:

```
./configure &&
make CC="gcc -static" &&
make BINDIR=$LFS/usr/bin \
   MANDIR=$LFS/usr/share/man/man1 install
```

Command explanations

make CC="gcc −static" This is used to build mawk statically.

Contents of mawk−1.3.3

Program Files

awk (link to mawk) and mawk

Descriptions

awk

awk is symlinked to mawk for programs which just look for any generic awk.

mawk

Mawk is an interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing, and for prototyping and experimenting with algorithms.

Dependencies

Mawk−1.3.3 needs the following to be installed:

bash: sh fileutils: chmod, cp, ln, rm binutils: as, ld diffutils: cmp gcc: cc1, collect2, cpp0, gcc
grep: egrep, grep make: make sed: sed sh−utils: hostname, tee textutils: cat, tr

Installing Patch−2.5.4

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Patch

Install Patch by running the following commands:

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

Command explanations

CPPFLAGS=-D_GNU_SOURCE: Adding **-D_GNU_SOURCE** to CPPFLAGS command before we configure patch fixes installation of the package on PPC and m68k platforms (that we know of). It also doesn't hurt compilation on other platforms (such as x86) so we do it by default.

Contents of patch−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 1MB 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 1MB or just as a patch file of 1KB 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.

Dependencies

Patch−2.5.4 needs the following to be installed:

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:           1 minute
Estimated required disk space:  2 MB
```

Installation of Sed

When installing Sed using glibc−2.1.x on your base system, it may be necessary to use a fix to prevent a variable name conflict. The following commands can be used in this case. Note that these commands can also be used for other glibc versions so if you aren't sure, then use the first version.

```
export CPPFLAGS=-Dre_max_failures=re_max_failures2 &&
./configure --prefix=$LFS/usr --bindir=$LFS/bin &&
unset CPPFLAGS &&
make LDFLAGS=-static &&
make install
```

If you are using a newer glibc version (2.2.x), you can use the following commands to install Sed:

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

Contents of sed−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).

Dependencies

Sed−3.02 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: echo, expr, hostname, sleep texinfo: install−info, makeinfo textutils: cat, tr

Installing Sh–utils–2.0

```
Estimated build time:           2 minutes
Estimated required disk space:  23 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.

Apply the patch by running the following command:

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

Install Sh–utils by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&
make LDFLAGS=-static &&
make install &&
cd $LFS/usr/bin &&
mv basename date echo false hostname $LFS/bin &&
mv pwd sleep stty test true uname $LFS/bin &&
mv chroot ../sbin
```

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, that 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 anyways, you can just pretend you didn't see it.

Contents of sh–utils–2.0

Program Files

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, hostname, 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 the groups a user is in.

hostid

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

hostname

hostname sets or prints the name of the current host system

id

id prints the real and effective UIDs and GIDs of a user or the current 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 non–tty

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 printf C 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's effective userid.

yes

yes outputs a string repeatedly until killed.

Dependencies

Sh−utils−2.0 needs the following to be installed:

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 mawk: mawk 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:           1 minute
Estimated required disk space:  7 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=$LFS/usr --disable-nls \
   --libexecdir=$LFS/usr/bin --bindir=$LFS/bin &&
make LDFLAGS=-static &&
make install
```

Contents of tar−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.

Dependencies

Tar−1.13 needs the following to be installed:

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
mawk: mawk net−tools: hostname patch: patch sed: sed sh−utils: basename, echo, expr, sleep, uname
texinfo: install−info, makeinfo textutils: cat, tr

Installing Texinfo−4.0

```
Estimated build time:           1 minute
Estimated required disk space:  11 MB
```

Installation of Texinfo

Install Texinfo by running the following commands:

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

Contents of texinfo−4.0

Program Files

info, 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 be more in depth than just explaining the options to a program.

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, it is also necessary to delete the topic in the index file as well. 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.

Dependencies

Texinfo−4.0 needs the following to be installed:

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−2.0

```
Estimated build time:           2 minutes
Estimated required disk space:  24 MB
```

Installation of Textutils

Install Textutils by running the following commands:

```
./configure --prefix=$LFS/usr --disable-nls &&
make LDFLAGS=-static &&
make install &&
mv $LFS/usr/bin/cat $LFS/usr/bin/head $LFS/bin
```

Contents of textutils−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

Print 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.

Dependencies

Textutils−2.0 needs the following to be installed:

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
mawk: mawk net−tools: hostname perl: perl sed: sed sh−utils: basename, echo, expr, sleep, uname
tar: tar texinfo: install−info, makeinfo textutils: cat, tr

Creating passwd and group files

In order for the user and group root to be recognized and to be able to login, there needs to be an entry in the /etc/passwd and /etc/group file. Besides the group root, a couple of other groups are recommended and needed by packages. The groups with their GID's below aren't part of any standard. The LSB only recommends a group bin with GID 1 to be present besides group root. Other group names and GID's can be chosen by the user. Well written packages don't depend on GID numbers but just use the group name, so it doesn't matter which GID a group has. Since there aren't any standards for groups the groups created here are the groups the MAKEDEV script (the script that creates the device files in the /dev directory) mentions.

Create a new file `$LFS/etc/passwd` by running the following command:

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

Create a new file `$LFS/etc/group` by running the following command:

```
cat > $LFS/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
```

Copying old NSS library files

If your normal Linux system runs Glibc−2.0, you need to copy the NSS library files to the LFS partition. Certain statically linked programs still depend on the NSS library, especially programs that need to lookup usernames, userid's and groupid's. You can check which C library version your normal Linux system uses by simply executing the library, like this:

```
/lib/libc.so.6
```

The first line will give you the release version. Following lines contain interesting information. If you have Glibc−2.0.x installed on your starting distribution, copy the NSS library files by running:

```
cp −av /lib/libnss* $LFS/lib
```

Mounting $LFS/proc file system

In order for certain programs to function properly, the proc file system must be mounted and available from within the chroot'ed environment as well. It's not a problem to mount the proc file system (or any other file system for that matter) twice or even more than that.

If you're still logged in as user "lfs", you should log out and log in again as user root. The reason for this is simple: only root is allowed to mount filesystems and to run chroot.

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

```
mount proc $LFS/proc -t proc
```

Chapter 6. Installing basic system software

Introduction

The installation of all the software is pretty straightforward and you will probably think it's so much easier and shorter to give the generic installation instructions for each package and only explain how to install something if a certain package requires an alternate installation method. Although I agree on that, I choose to give the full instructions for each and every package. This is simply to avoid any possible confusion and errors.

Now would be a good time to take a look at the optimization hint at http://hints.linuxfromscratch.org/hints/optimization.txt if you plan on using compiler optimization for the packages installed in the following chapter. Compiler optimization can make a program run faster, but may also cause some compilation problems. If you run into problems after having used optimization, always try it without optimizing to see if the problem persists.

About debugging symbols

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

A program compiled with debugging symbols means a user can run a program or library through a debugger and the debugger's output will be user friendly. These debugging symbols also enlarge the program or library significantly.

Before you start wondering whether these debugging symbols really make a big difference, here are some statistics. Use them to draw your own conclusion.

- A dynamic Bash binary with debugging symbols: 1.2MB
- A dynamic Bash binary without debugging symbols: 478KB
- /lib and /usr/lib (glibc and gcc files) with debugging symbols: 87MB
- /lib and /usr/lib (glibc and gcc files) without debugging symbols: 16MB

Sizes vary depending on which compiler was used and which C library version was used to link dynamic programs against, but results will be similar if you compare programs with and without debugging symbols.

To remove debugging symbols from a binary (must be an a.out or ELF binary) run **`strip --strip-debug filename`**. Wildcards can be used to strip debugging symbols from multiple files (use something like **`strip --strip-debug $LFS/usr/bin/*`**). Most people will probably never use a debugger on software, so by removing those symbols a lot of disk space can be regained.

For your convenience, chapter 9 includes one simple command to strip all debugging symbols from all programs and libraries on your system.

You might find additional information in the optimization hint which can be found at http://hints.linuxfromscratch.org/hints/optimization.txt .

Creating $LFS/root/.bash_profile

When we have entered the chroot'ed environment in the next section we want to export a couple of environment variables in that shell such as PS1, PATH and others variables which are good to have set. For that purpose we'll create the $LFS/root/.bash_profile file which will be read by bash when we enter the chroot environment.

Create a new file `$LFS/root/.bash_profile` by running the following.

```
cat > $LFS/root/.bash_profile << "EOF"
# Begin /root/.bash_profile

PS1='\u:\w\$ '
PATH=/bin:/usr/bin:/sbin:/usr/sbin

export PS1 PATH

# End /root/.bash_profile
EOF
```

The PS1 variable is an environment variable that controls the appearance of the command prompt. See the bash man page for details how this variable is constructed. Additional environment variables, aliases and so forth that are needed and/or wanted can be added at your own discretion.

Entering the chroot'ed environment

It's time to enter our chroot'ed environment in order to install the rest of the software we need.

Enter the following commands to enter the chroot'ed environment. From this point on there's no need to use the $LFS variable anymore, because everything a user does will be restricted to the LFS partition (since / is actually /mnt/lfs but the shell doesn't know that).

```
cd $LFS &&
chroot $LFS /usr/bin/env -i HOME=/root \
   TERM=$TERM /bin/bash --login
```

The −i option will clear all environment variables for as long as you are in the chroot'ed environment and only the HOME and TERM variables are set. The TERM=$TERM construction will set the TERM variable inside chroot to the same value as outside chroot which is needed for programs like vim and less to operate properly. If you need other variables present, such as CFLAGS or CXXFLAGS, you need to set them again.

The reason we do **`cd $LFS`** before running the **`chroot`** command is that older sh−utils packages have a chroot program which doesn't do the cd by itself, therefore meaning that we have to perform it manually. While this isn't an issue with most modern distributions, it does no harm anyways and ensures that the command works for everyone.

Now that we are inside a chroot'ed environment, we can continue to install all the basic system software. You have to make sure all the following commands in this and following chapters are run from within the chroot'ed environment. If you ever leave this environment for any reason (when rebooting for example) please remember to mount $LFS/proc again and re−enter chroot before continuing with the book.

Note that the bash prompt will contain "I have no name!" This is normal because Glibc hasn't been installed yet.

Dependencies

Chroot needs the following to be installed:

bash: bash sh−utils: env

Changing ownership of the LFS partition

Now we're in chroot, it is a good time to change the ownership of all files and directories that were installed in chapter 5 back to root. Run the following commands to do so:

```
cd / &&
chown 0.0 . proc &&
chown -R 0.0 bin boot dev etc home lib mnt opt root sbin tmp usr var
```

Depending on the filesystem you created on the LFS partition, you may have a /lost+found directory. If so, run:

```
chown 0.0 lost+found
```

These commands will change the ownership of the root partition and the `/proc` directory to root, plus everything under the directories mentioned in the second line. In these commands, 0.0 is used instead of the usual root.root, because the username root can't be resolved because glibc is not yet installed.

Creating the /etc/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 -s /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 /etc/mtab will always be up−to−date.

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

Installing Glibc−2.2.5

```
Estimated build time:           46 minutes
Estimated required disk space:  350 MB
```

Installation of Glibc

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, so we recommend you unsetting CFLAGS, CXXFLAGS and other such variables/settings that would change the default optimization that it comes with. 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:

```
mknod -m 0666 /dev/null c 1 3 &&
touch /etc/ld.so.conf &&
cp malloc/Makefile malloc/Makefile.backup &&
sed 's%\$(PERL)%/usr/bin/perl%' malloc/Makefile.backup > malloc/Makefile &&
cp login/Makefile login/Makefile.backup &&
sed 's/root/0/' login/Makefile.backup > login/Makefile &&
mkdir ../glibc-build &&
cd ../glibc-build &&
../glibc-2.2.5/configure --prefix=/usr \
   --enable-add-ons --libexecdir=/usr/bin &&
echo "cross-compiling = no" > configparms &&
make &&
make install &&
make localedata/install-locales &&
exec /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.

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) is not fatal. The files msgfmt would create are already pre−built, so you won't be missing out on anything. You would only need it if you make changes to the Glibc internationalization (i18n) files. Since we don't do this by default, we can safely ignore it.

Command explanations

mknod −m 0666 /dev/null c 1 3: Glibc needs a null device to compile properly. All other devices will be created in the next section.

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).

sed 's%\$(PERL)%/usr/bin/perl%' malloc/Makefile.backup > malloc/Makefile: This sed command searches through `malloc/Makefile.backup` and converts all occurrences of `$(PERL)` to `/usr/bin/perl`. The output is then written to the original `malloc/Makefile.in` which is used during configuration. This is done because Glibc can't autodetect perl since it hasn't been installed yet.

sed 's/root/0' login/Makefile.backup > login/Makefile: This sed command replaces all occurrences of `root` in `login/Makefile.backup` with 0. This is because we don't have glibc on the LFS system yet, so usernames can't be resolved to their user id's. Therefore, we replace the username root with user id 0.

−−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 userid to a username (which in turn was caused by the missing Glibc installation).

Contents of glibc−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

No description is currently available.

gencat

gencat generates message catalogues.

getconf

No description is currently available.

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

No description is currently available.

locale

No description is currently available.

localedef

localedef compiles locale specifications.

mtrace

No description is currently available.

nscd

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

nscd_nischeck

No description is currently available.

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

No description is currently available.

rpcinfo

No description is currently available.

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

No description is currently available.

libSegFault

No description is currently available.

libanl, libanl_p

No description is currently available.

libbsd−compat

No description is currently available.

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) come in two flavors: dynamic ones and static ones. In short when a program uses a static C library, the code from the C library will be copied into the executable file. When a program uses a dynamic library, that 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

No description is currently available.

libg

No description is currently available.

libieee

No description is currently available.

libm, libm_p

libm is the mathematical library.

libmcheck

No description is currently available.

libmemusage

No description is currently available.

libnsl, libnsl_p

No description is currently available.

libnss_compat, libnss_dns, libnss_files, libnss_hesiod, libnss_nis, libnss_nisplus

No description is currently available.

libpcprofile

No description is currently available.

libpthread, libpthread_p

No description is currently available.

libresolv, libresolv_p

No description is currently available.

librpcsvc, librpcsvc_p

No description is currently available.

librt, librt_p

No description is currently available.

libthread_db

No description is currently available.

libutil, libutil

No description is currently available.

Dependencies

Glibc−2.2.5 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: date, expr, hostname, pwd, uname texinfo: install−info, makeinfo textutils: cat, cut, sort, tr

Creating devices (Makedev−1.4)

```
Estimated build time:           1 minute
Estimated required disk space:  57 KB
```

Creating devices

Note: the MAKEDEV−1.4.bz2 file you have unpacked is not an archive, so it won't create a directory for you to cd into.

Create the device files by running the following commands:

```
cp MAKEDEV-1.4 /dev/MAKEDEV &&
cd /dev &&
chmod 754 MAKEDEV
```

Now, depending on whether you are going to use devpts or not, you can run one of two commands:

If you do not intend to use devpts, run:

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

If you do intend to use devpts, then run:

```
./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 the devices you need. If you are sure you are going to use devpts however, the other command makes sure that you don't create a set of devices which you don't require.

MAKEDEV will create hda[1−20] to hdh[1−20] and such but keep in mind that you may not be able to use all of those devices due to kernel limitations regarding the max. number of partitions.

Command explanations

`./MAKEDEV -v generic:` This creates generic devices. Normally, these devices are all the devices you need. It's possible that you are missing some special devices that are needed for your hardware configuration. Create them with ./MAKEDEV −v <device>. The **`generic-nopty`** option does a similar job but skips some devices which are not needed if you are using devpts.

Contents of MAKEDEV−1.4

Program Files

MAKEDEV

Descriptions

MAKEDEV

MAKEDEV is a script that can help in creating the necessary static device files that usually reside in the /dev directory. More information on device nodes can be found in the Linux Kernel source tree in `Documentation/devices.txt`.

Dependencies

MAKEDEV−1.4 needs the following to be installed:

bash: sh fileutils: chmod, chown, cp, ln, mknod, mv, rm grep: grep sh−utils: expr, id

Installing Man−pages−1.47

```
Estimated build time:           1 minute
Estimated required disk space:  5 MB
```

Installation of Man−pages

Install Man−pages by running the following commands:

```
make install
```

Contents of manpages−1.47

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.

Dependencies

Man−pages−1.47 needs the following to be installed:

bash: sh fileutils: install make: make

Installing Findutils−4.1

```
Estimated build time:           1 minute
Estimated required disk space:  3 MB
```

Installing Findutils

Before Findutils is installed the findutils patch file has to be unpacked.

Install Findutils by running the following commands:

```
patch −Np1 −i ../findutils−4.1.patch &&
./configure −−prefix=/usr &&
make &&
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 &&
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−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

updatedb runs a program called frcode 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 contain 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 system 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 file can be created that contains all these files (one per line) and use xargs to perform that command on the list.

Dependencies

Findutils−4.1 needs the following to be installed:

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 Mawk−1.3.3

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Mawk

Install Mawk by running the following commands:

```
./configure &&
make &&
make BINDIR=/usr/bin \
   MANDIR=/usr/share/man/man1 install &&
cd /usr/bin &&
ln −sf mawk awk
```

Contents of mawk−1.3.3

Program Files

awk (link to mawk) and mawk

Descriptions

awk

awk is symlinked to mawk for programs which just look for any generic awk.

mawk

Mawk is an interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing, and for prototyping and experimenting with algorithms.

Dependencies

Mawk−1.3.3 needs the following to be installed:

bash: sh fileutils: chmod, cp, ln, rm binutils: as, ld diffutils: cmp gcc: cc1, collect2, cpp0, gcc grep: egrep, grep make: make sed: sed sh−utils: hostname, tee textutils: cat, tr

Installing Ncurses–5.2

```
Estimated build time:           6 minutes
Estimated required disk space:  29 MB
```

Installation of Ncurses

Install Ncurses by running the following commands:

```
./configure --prefix=/usr --libdir=/lib \
   --with-shared --disable-termcap &&
make &&
make install &&
cd /lib &&
mv *.a /usr/lib &&
chmod 755 *.5.2 &&
cd /usr/lib &&
ln -sf libncurses.a libcurses.a &&
ln -sf ../../lib/libncurses.so &&
ln -sf ../../lib/libcurses.so &&
ln -sf ../../lib/libform.so &&
ln -sf ../../lib/libpanel.so &&
ln -sf ../../lib/libmenu.so
```

Command explanations

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

`--disable-termcap:` Disabled the compilation of termcap fall back support.

`cd /lib && mv *.a /usr/lib :` This moves all of the static ncurses library files from /lib to /usr/lib. /lib should only contain the shared files which are essential to the system when /usr may not be mounted.

`chmod 755 *.5.2:` Shared libraries should be executable. Ncurses 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.

`ln -sf ../../lib/libncurses.so etc:` These symlinks are created so the linker will use the dynamic versions of these libraries instead of the static ones, which happens if the linker can't find the dynamic versions.

Contents

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

The libraries that make up the Ncurses library 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. The libncurses libraries are the base of the system.

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.

Dependencies

Ncurses−5.2 needs the following to be installed:

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
mawk: mawk sed: sed sh−utils: basename, date, echo, expr, hostname, uname
textutils: cat, sort, tr, wc

Installing Vim−6.0

```
Estimated build time:           2 minutes
Estimated required disk space:  15 MB
```

Installation of Vim

If you don't like vim to be installed as an editor on the LFS system, you may want to download an alternative and install an editor you prefer. There are a few hints how to install different editors available at http://hints.linuxfromscratch.org/hints/. The hints which are currently available are for Emacs, Joe and nano.

Install Vim by running the following commands:

```
cp runtime/syntax/sh.vim runtime/syntax/sh.vim.backup &&
sed '/shUntil\|link shRepeat/{
   /shUntil/N
   /^/i\
   if exists("b:is_kornshell") || exists("b:is_bash")
   p
   /$/i\
   endif
   d
   }' runtime/syntax/sh.vim.backup > runtime/syntax/sh.vim &&
./configure --prefix=/usr &&
make CPPFLAGS=-DSYS_VIMRC_FILE=\\\"/etc/vimrc\\\" &&
make install &&
cd /usr/bin &&
ln -sf vim vi
```

If you plan on installing 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.

FHS compliance notes

The FHS says that editors like vim should use /var/lib/<editor> for their temporary state files, like temporary save files for example. If you wish vim to conform to the FHS, you should use this command set instead of the one presented above:

```
cp runtime/syntax/sh.vim runtime/syntax/sh.vim.backup &&
sed '/shUntil\|link shRepeat/{
   /shUntil/N
   /^/i\
   if exists("b:is_kornshell") || exists("b:is_bash")
   p
   /$/i\
   endif
   d
   }' runtime/syntax/sh.vim.backup > runtime/syntax/sh.vim &&
./configure --prefix=/usr --localstatedir=/var/lib/vim &&
make CPPFLAGS=-DSYS_VIMRC_FILE=\\\"/etc/vimrc\\\" &&
make install &&
cd /usr/bin &&
ln -sf vim vi
```

Command explanations

sed '/shUntil\|link shRepeat/{...: This sed command fixes a bug in the `syntax/sh.vim` file that will cause an error message when you edit a shell script using syntax highlighting.

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

Program Files

ex (link to vim), rview (link to vim), rvim (link to vim), vi (link to vim), view (link to vim), vim, vimdiff (link to vim), vimtutor (link to vim) and xxd

Descriptions

ex

ex starts vim in Ex mode.

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.

vi

vi starst vim in vi–compatible mode.

view

view starts vim in read–only mode.

vim

vim starts vim in the normal, default way.

vimdiff

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

vimtutor

vimtutor starts the Vim tutor.

xxd

xxd makes a hexdump or does the reverse.

Dependencies

Vim−6.0 needs the following to be installed:

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 GCC−2.95.3

```
Estimated build time:           22 minutes
Estimated required disk space:  148 MB
```

Installation of GCC

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, so we recommend you unsetting CFLAGS, CXXFLAGS and other such variables/settings that would change the default optimization that it comes with.

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-2.95.3-2.patch &&
mkdir ../gcc-build &&
cd ../gcc-build &&
../gcc-2.95.3/configure --prefix=/usr --enable-shared \
   --enable-languages=c,c++ --enable-threads=posix &&
make bootstrap &&
make install &&
cd /lib &&
ln -sf ../usr/bin/cpp &&
cd /usr/lib &&
ln -sf ../bin/cpp &&
cd /usr/bin &&
ln -sf gcc cc
```

Contents of gcc−2.95.3

Program Files

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

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

c++filt is used to demangle C++ symbols.

collect2

No description is currently available.

cpp, cpp0

cpp pre–processes a source file, such as including the contents of header files into the source file. It's a good idea to not do this manually to save a lot of time. Someone just inserts a line like #include <filename>. The preprocessor inserts the contents of that file into the source file. That's one of the things a preprocessor does.

gcov

No description is currently available.

protoize

No description is currently available.

unprotoize

No description is currently available.

Library Files

libgcc.a, libiberty.a, libstdc++.[a,so]

libgcc

libgcc.a is a run–time support file for gcc.Most of the time, on most machines, libgcc.a is not actually necessary.

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.

Dependencies

GCC–2.95.3 needs the following to be installed:

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 Bison−1.31

```
Estimated build time:           1 minute
Estimated required disk space:  3 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 yacc script 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−1.31

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

Dependencies

Bison−1.31 needs the following to be installed:

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−358

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Less

Install Less by running the following commands:

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

Contents of less−358

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 with 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.

Dependencies

Less−358 needs the following to be installed:

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−1.17.2

```
Estimated build time:           2 minutes
Estimated required disk space:  16 MB
```

Installation of Groff

Install Groff by running the following commands:

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

Contents of groff−1.17.2

Program Files

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

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.

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.

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.

Dependencies

Groff−1.17.2 needs the following to be installed:

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 mawk: awk sed: sed
sh−utils: basename, date, echo, expr, hostname, uname textutils: cat, tr

Installing Man−1.5j

```
Estimated build time:           1 minute
Estimated required disk space:  1 MB
```

Installation of Man

Run the following commands to install man:

```
./configure −default &&
make &&
make install
```

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.

Contents of man−1.5j

Program Files

apropos, makewhatis, man, man2dvi, man2html and whatis

Descriptions

apropos

apropos searches a set of database files containing short descriptions of system commands for keywords 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 a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output. Only complete word matches are displayed.

Dependencies

Man−1.5i2 needs the following to be installed:

bash: sh binutils: as, ld fileutils: chmod, cp, install, mkdir, rm gcc: c11, collect2, cpp0, gcc
grep: grep make: make mawk: awk sed: sed sh−utils: echo textutils: cat

Installing Perl−5.6.1

```
Estimated build time:           6 minutes
Estimated required disk space:  35 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 build, 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−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

pos2usage 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.

Dependencies

Perl−5.6.1 needs the following to be installed:

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 mawk: awk sed: sed
sh−utils: basename, date, echo, expr, hostname, pwd, uname, whoami
textutils: cat, comm, sort, split, tr, uniq, wc

Installing M4−1.4

```
Estimated build time:           1 minute
Estimated required disk space:  3 MB
```

Installation of M4

Install M4 by running the following commands:

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

Contents of m4−1.4

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. M4 can be used either as a front−end to a compiler or as a macro processor in its own right.

Dependencies

M4−1.4 needs the following to be installed:

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−4.0

```
Estimated build time:           1 minute
Estimated required disk space:  10 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−4.0

Program Files

info, 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 be more in depth than just explaining the options to a program.

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, it is also necessary to delete the topic in the index file as well. 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.

Dependencies

Texinfo−4.0 needs the following to be installed:

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−2.52

```
Estimated build time:           1 minute
Estimated required disk space:  4 MB
```

Installation of Autoconf

Autoconf−2.52 is known to be too new for some applications. KDE−CVS is mostly reported not to work well with this automake release and downgrading to version 2.13 is recommended if you start experiencing any problems with this release.

Install Autoconf by running the following commands:

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

Contents of autoconf−2.52

Program Files

autoconf, autoheader, 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

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 source files in the directory tree rooted at a directory given as a command line argument, or the current directory if none is given. It searches the source files for common portability problems and creates a file configure.scan which is a 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 figure out what its configure needs to check for. It may help fill in some gaps in a configure.in generated by autoscan.

Dependencies

Autoconf−2.52 needs the following to be installed:

bash: sh diffutils: cmp fileutils: chmod, install, ln, ls, mkdir, mv, rm grep: fgrep, grep m4: m4
make: make mawk: mawk sed: sed sh−utils: echo, expr, hostname, sleep, uname texinfo: install−info
textutils: cat, tr

Installing Automake−1.5

```
Estimated build time:           1 minute
Estimated required disk space:  3 MB
```

Installation of Automake

Automake−1.5 is known to be too new for some applications. KDE−CVS is mostly reported not to work well with this automake release and downgrading to version 1.4−p5 is recommended if you start experiencing any problems with this release.

Install Automake by running the following commands:

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

Contents of automake−1.5

Program Files

aclocal and automake

Descriptions

aclocal

Automake includes a number of Autoconf macros which can be used in packages; some of them are actually required by Automake in certain situations. These macros must be defined in the aclocal.m4−file; otherwise 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

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.

Dependencies

Automake−1.5 needs the following to be installed:

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:           3 minutes
Estimated required disk space:  19 MB
```

Installation of Bash

Install Bash by running the following commands:

```
./configure --prefix=/usr --with-curses \
   --bindir=/bin &&
make &&
make install &&
cd /bin &&
ln -sf bash sh &&
exec /bin/bash --login
```

Contents of bash−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. Bash is a program that 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.

Dependencies

Bash−2.05a needs the following to be installed:

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 mawk: awk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info textutils: cat, tr, uniq

Installing Flex−2.5.4a

```
Estimated build time:           1 minute
Estimated required disk space:  3MB
```

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 lex script 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−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 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 sets up rules for what to look for than to write the actual program that finds the text.

flex++

flex++ invokes a version of flex which is used exclusively for C++ scanners.

lex

We create a yacc script 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

No description is currently available.

Dependencies

Flex−2.5.4a needs the following to be installed:

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 File−3.37

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of File

Install File by running the following commands:

```
touch aclocal.m4 configure Makefile.in stamp-h.in &&
./configure --prefix=/usr --datadir=/usr/share/misc &&
make &&
make install
```

Command explanations

`touch aclocal.m4 configure Makefile.in stamp-h.in`: This command works around an error which occurs when compiling file with automake−1.5 installed by changing the modification dates of some files to the current date. Changing the date will cause make to think the files are already up−to−date so they're not recreated.

Contents of file−3.37

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.

Dependencies

File−3.37 needs the following to be installed:

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 mawk: mawk sed: sed sh−utils: echo, expr, hostname, sleep texinfo: makeinfo textutils: cat, tr

Installing Libtool−1.4.2

```
Estimated build time:           1 minute
Estimated required disk space:  5 MB
```

Installation of Libtool

Install Libtool by running the following commands:

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

Contents of libtool−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,so]

Descriptions

libltdl

Libtool provides a small library, called `libltdl', that aims at hiding the various difficulties of dlopening libraries from programmers.

Dependencies

Libtool−1.4.2 needs the following to be installed:

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 Bin86−0.16.0

```
Estimated build time:           1 minute
Estimated required disk space:  1 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 bin86−0.16.0

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

No description is currently available.

objdump86

No description is currently available.

size86

No description is currently available.

Dependencies

Bin86–0.16.0 needs the following to be installed:

bash: sh binutils: as, ld, strip fileutils: chmod, install, ln, mv gcc: cc, cc1, collect2, cpp0
make: make sed: sed

Installing Binutils−2.11.2

```
Estimated build time:           6 minutes
Estimated required disk space:  85 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, so we recommend you unsetting CFLAGS, CXXFLAGS and other such variables/settings that would change the default optimization that it comes with.

Install Binutils by running the following commands:

```
mkdir ../binutils-build &&
cd ../binutils-build &&
../binutils-2.11.2/configure --prefix=/usr --enable-shared &&
make tooldir=/usr &&
make tooldir=/usr install &&
make tooldir=/usr install-info
```

Command explanations

make tooldir=/usr install-info: This will install binutil's info pages.

Contents of binutils−2.11.2

Program Files

addr2line, ar, as, c++filt, 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.

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.

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 a member of an archive 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 objfile 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, libiberty.a and libopcodes.a

Descriptions

libbfd

libbfd is the Binary File Descriptor library.

libiberty

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

libopcodes

No description is currently available.

Dependencies

Binutils−2.11.2 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, true, uname texinfo: install−info, makeinfo
textutils: cat, sort, tr, uniq

Installing Bzip2−1.0.1

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Bzip2

Install Bzip2 by running the following commands:

```
make −f Makefile−libbz2_so &&
make bzip2recover libbz2.a &&
ln −s libbz2.so.1.0.1 libbz2.so &&
cp bzip2−shared /bin/bzip2 &&
cp bzip2recover /bin &&
cp bzip2.1 /usr/share/man/man1 &&
cp bzlib.h /usr/include &&
cp −a libbz2.so* /lib &&
rm /usr/lib/libbz2.a &&
cp libbz2.a /usr/lib &&
cd /usr/lib &&
ln −sf ../../lib/libbz2.so &&
cd /bin &&
ln −sf bzip2 bunzip2 &&
ln −sf bzip2 bzcat &&
cd /usr/share/man/man1 &&
ln −sf bzip2.1 bunzip2.1 &&
ln −sf bzip2.1 bzcat.1 &&
ln −sf bzip2.1 bzip2recover.1
```

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 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.

The reason we don't use **`make install`** is that bzip2's make install doesn't install the shared libbz2.so, nor the bzip2 binary that's linked against that library. So we have no choice but to manually install the files.

Contents of bzip2−1.0.1

Program Files

bunzip2 (link to bzip2), bzcat (link to bzip2), bzip2 and bzip2recover

Descriptions

bunzip2

Bunzip2 decompresses files that are compressed with bzip2.

bzcat

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

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.

Library Files

libbz2.[a,so]

libbz2

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

Dependencies

Bzip2−1.0.1 needs the following to be installed:

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:           1 minute
Estimated required disk space:  2 MB
```

Installation of Ed

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:

```
cp buf.c buf.c.backup &&
sed 's/int u/int u, sfd/' buf.c.backup | \
   sed '/.*\*mktemp.*/d' | \
   sed 's/.*if (mktemp.*/  sfd = mkstemp(sfn);\
   if ((sfd == -1) || (sfp = fopen (sfn, "w+")) == NULL)/' > buf.c &&
./configure --prefix=/usr &&
make &&
make install &&
mv /usr/bin/ed /usr/bin/red /bin
```

Command explanations

The sed commands fix 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 various files, etc.

Contents of ed−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.

Dependencies

Ed−0.2 needs the following to be installed:

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 Gettext−0.10.40

```
Estimated build time:           1 minute
Estimated required disk space:  11MB
```

Installation of Gettext

Install Gettext by running the following commands:

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

Contents of gettext−0.10.40

Program Files

gettext, gettextize, msgcmp, msgcomm, msgfmt, msgmerge, msgunfmt, ngettext and xgettext

Descriptions

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 users native language rather than in the default English language.

gettextize

No description is currently available.

msgcmp

No description is currently available.

msgcomm

No description is currently available.

msgfmt

No description is currently available.

msgmerge

No description is currently available.

msgunfmt

No description is currently available.

ngettext

No description is currently available.

xgettext

No description is currently available.

Dependencies

Gettext−0.10.40 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info, makeinfo
textutils: cat, sort, tr, uniq

Installing Kbd−1.06

```
Estimated build time:           1 minute
Estimated required disk space:  8 MB
```

Installation of Kbd

Install Kbd by running the following commands:

```
patch −Np1 −i ../kbd−1.06−2.patch &&
./configure &&
make &&
make install
```

Command explanations

`patch −Np1 −i ../kbd−1.06−2.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 (setlogcons, setvesablank and getunimap) that are not installed by default, are installed as well.

Contents of kbd−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 it is by using this program that 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.

Dependencies

Kbd−1.06 needs the following to be installed:

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−2.7

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Diffutils

Install Diffutils by running the following commands:

```
./configure --prefix=/usr &&
make PR_PROGRAM=/usr/bin/pr &&
make install
```

Contents of diffutils−2.7

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.

Dependencies

Diffutils−2.7 needs the following to be installed:

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−1.25

```
Estimated build time:           2 minutes
Estimated required disk space:  21 MB
```

Installation of E2fsprogs

Install E2fsprogs by running the following commands:

```
mkdir ../e2fsprogs-build &&
cd ../e2fsprogs-build &&
../e2fsprogs-1.25/configure --prefix=/usr --with-root-prefix="" \
   --enable-elf-shlibs &&
make &&
make install &&
make install-libs
```

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' 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−1.25

Program Files

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, 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 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 is used to check and optionally repair Linux second extended filesystems. fsck.ext2 does the same as e2fsck.

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

No description is currently available.

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.

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 among UUIDs created 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

No description is currently available.

libe2p

No description is currently available.

libext2fs

No description is currently available.

libss

No description is currently available.

libuuid

No description is currently available.

Dependencies

E2fsprogs−1.25 needs the following to be installed:

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 mawk: awk sed: sed
sh−utils: basename, echo, expr, hostname, uname texinfo: makeinfo textutils: cat, tr

Installing Fileutils−4.1

```
Estimated build time:           3 minutes
Estimated required disk space:  16 MB
```

Installation of Fileutils

Install Fileutils by running the following commands:

```
./configure --prefix=/usr --bindir=/bin &&
make &&
make install &&
cd /usr/bin &&
ln -sf ../../bin/install
```

Contents of fileutils−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 argument and for each subdirectory of directory arguments.

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.

Dependencies

Fileutils−4.1 needs the following to be installed:

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 Grep–2.4.2

```
Estimated build time:           1 minute
Estimated required disk space:  3 MB
```

Installation of Grep

Install Grep by running the following commands:

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

Contents of grep–2.4.2

Program Files

egrep, fgrep 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.

Dependencies

Grep–2.4.2 needs the following to be installed:

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
mawk: mawk 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:           1 minute
Estimated required disk space:  1 MB
```

Installation of Gzip

Install Gzip by running the following commands:

```
./configure --prefix=/usr &&
cp gzexe.in gzexe.in.backup &&
sed 's%"BINDIR"%/bin%' gzexe.in.backup > gzexe.in &&
make &&
make install &&
cd /usr/bin &&
mv gzip /bin &&
rm gunzip zcat &&
cd /bin &&
ln -sf gzip gunzip &&
ln -sf gzip zcat &&
ln -sf gunzip uncompress
```

Contents of gzip−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 either a list of files on the command line or its standard input and writes the uncompressed data on standard output

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.

Dependencies

Gzip−1.2.4a needs the following to be installed:

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 Lilo−22.1

```
Estimated build time:           1 minute
Estimated required disk space:  3 MB
```

Installation of Lilo

We have chosen Lilo because we feel comfortable with it, but you may wish to take a look elsewhere. Someone has written a hint on GRUB at http://hints.linuxfromscratch.org/hints/grub−howto.txt, an alternative boot loader.

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 being 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.

Maybe you'll be interested to know that someone wrote a hint on how to get a logo instead the the standard LILO prompt or menu. Take a look at it at http://hints.linuxfromscratch.org/hints/bootlogo.txt .

Contents of lilo−22.1

Program Files

lilo and mkrescue

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.

Dependencies

Lilo−22.1 needs the following to be installed:

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:           1 minute
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−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.

Dependencies

Make−3.79.1 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info, makeinfo
textutils: cat, tr

Installing Modutils−2.4.12

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Modutils

Install Modutils by running the following commands:

```
./configure &&
make &&
make install
```

Contents of modutils−2.4.12

Program Files

depmod, genksyms, insmod, insmod_ksymoops_clean, kallsyms (link to insmod), kernelversion, ksyms, lsmod (link to insmod), modinfo, modprobe (link to insmod) and rmmod

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.

Dependencies

Modutils−2.4.12 needs the following to be installed:

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:           1 minute
Estimated required disk space:  1 MB
```

Installation of Netkit−base

Install Netkit−base by running the following commands:

```
./configure &&
make &&
make install &&
cd etc.sample &&
cp services protocols /etc
```

There are other files in the `etc.sample` directory which might be of interest to you.

Contents of netkit−base−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.

Dependencies

Netkit−base−0.17 needs the following to be installed:

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:           1 minute
Estimated required disk space:  2 MB
```

Installation of Patch

Install Patch by running the following commands:

```
export CPPFLAGS=-D_GNU_SOURCE &&
./configure --prefix=/usr &&
unset CPPFLAGS &&
make &&
make install
```

Contents of patch−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 1MB 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 1MB or just as a patch file of 1KB 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.

Dependencies

Patch−2.5.4 needs the following to be installed:

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:           1 minute
Estimated required disk space:  170 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−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.

Dependencies

Procinfo−18 needs the following to be installed:

binutils: as, ld fileutils: install, mkdir gcc: cc1, collect2, cpp0, gcc make: make

Installing Procps−2.0.7

```
Estimated build time:           1 minute
Estimated required disk space:  2 MB
```

Installation of Procps

Install Procps by running the following commands:

```
make &&
make XSCPT='' install &&
mv /usr/bin/kill /bin
```

Command explanations

`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−2.0.7

Program Files

free, kill, oldps, pgrep, pkill, ps, skill, snice, sysctl, tload, top, uptime, 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.

uptime

uptime gives a one line display of the following information: the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

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.

Dependencies

Procps−2.0.7 needs the following to be installed:

bash: sh binutils: as, ld, strip fileutils: install, ln, mv, rm gcc: cc1, collect2, cpp0, gcc grep: grep make: make mawk: awk sed: sed sh−utils: basename, pwd textutils: sort, tr

Installing Psmisc−20.2

```
Estimated build time:           1 minute
Estimated required disk space:  500 KB
```

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 that isn't a problem because we install the sysvinit package later on which provides us with 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:

```
cd /bin
ln -s killall 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−20.2

Program Files

fuser, killall, pidof (link to 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 using the specified files or file systems.

killall

killall sends a signal to all processes running any of the specified commands.

pidof

Pidof finds the process id's (pids) of the named programs and prints those id's on standard output.

pstree

pstree shows running processes as a tree.

Dependencies

Psmisc−20.2 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: makeinfo textutils: cat, tr

Installing Reiserfsprogs–3.x.0j

```
Estimated build time:           TBD
Estimated required disk space:  TBD
```

Installation of Reiserfsprogs

Reiserfsprogs only needs to be installed if you intend on using the reiserfs filesystem. Install Reiserfsprogs by running the following commands:

```
./configure --mandir=/usr/share/man &&
make &&
make install
```

Command explanations

--mandir=/usr/share/man: This ensures that the manual pages are installed in the correct location while still installing the programs in `/sbin` as they should be.

Contents of reiserfsprogs–3.x.0j

Program Files

debugreiserfs, mkreiserfs, reiserfsck, resize_reiserfs and unpack

Descriptions

debugreiserfs

debugreiserfs can sometimes help to solve problems with reiserfs filesystems. If it is called without options it prints the super block of any reiserfs filesystem found on the device.

mkreiserfs

mkreiserfs creates a reiserfs file system.

reiserfsck

reiserfsck checks a reiserfs file system.

resize_reiserfs

resize_reiserfs is used to resize an unmounted reiserfs file system

unpack

No description is currently available.

Dependencies

Reiserfs−3.x.0j needs the following to be installed:

autoconf: autoconf, autoheader automake: aclocal, automake bash: sh binutils: ar, as, ld, ranlib diffutils: cmp fileutils: chmod, install, ls, rm gcc: cc1, collect2, cpp0, gcc grep: egrep, grep m4: m4 make: make mawk: mawk sed: sed sh−utils: echo, expr, hostname, sleep texinfo: makeinfo textutils: cat, tr

Installing Sed−3.02

```
Estimated build time:           1 minute
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−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).

Dependencies

Sed−3.02 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: echo, expr, hostname, sleep texinfo: install−info, makeinfo textutils: cat, tr

Installing Sh−utils−2.0

```
Estimated build time:           2 minutes
Estimated required disk space:  11 MB
```

Installation of Sh−utils

Install Shellutils by running the following commands:

```
./configure --prefix=/usr &&
make &&
make install &&
cd /usr/bin &&
mv basename date echo false hostname /bin &&
mv pwd sleep stty su test true uname /bin &&
mv chroot ../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. We create that in this way, while still in the `/usr/bin` directory:

```
cd /bin &&
ln -sf test [
```

Contents of sh−utils−2.0

Program Files

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, hostname, 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 the groups a user is in.

hostid

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

hostname

hostname sets or prints the name of the current host system

id

id prints the real and effective UIDs and GIDs of a user or the current 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 non–tty

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 printf C 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's effective userid.

yes

yes outputs a string repeatedly until killed.

Dependencies

Sh−utils−2.0 needs the following to be installed:

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 mawk: mawk perl: perl sed: sed sh−utils: basename, echo, expr, hostname, sleep, uname tar: tar texinfo: install−info, makeinfo textutils: cat, tr

Installing Net–tools–1.60

```
Estimated build time:           1 minute
Estimated required disk space:  5 MB
```

Installation of Net–tools

Install Net–tools by running the following commands:

```
make &&
make update
```

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–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 is used to set or show the system's hostname

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

updomainname shows or sets the system's NIS/YP domain name.

Dependencies

Net−tools−1.60 needs the following to be installed:

bash: bash, sh binutils: ar, as, ld fileutils: install, ln, ls, mv, rm gcc: cc, cc1, collect2, cpp0
make: make sh−utils: echo

Installing Shadow−20001016

```
Estimated build time:           3 minutes
Estimated required disk space:  6 MB
```

Installation of Shadow Password Suite

Before you install this package, you may want to have a look at the http://hints.linuxfromscratch.org/hints/shadowpasswd_plus.txt lfs hint. It discusses how you can make your system more secure regarding passwords and how to get the most out of this Shadow package.

Install the Shadow Password Suite by running the following commands:

```
cp src/useradd.c src/useradd.c.backup &&
sed 's/\(.*\) (nflg || \(.*\))\(.*\)/\1\2\3/' \
   src/useradd.c.backup > src/useradd.c &&
./configure --prefix=/usr &&
make &&
make install &&
cd etc &&
cp limits login.access /etc &&
sed 's%/var/spool/mail%/var/mail%' login.defs.linux > /etc/login.defs &&
cd /lib &&
mv libshadow.*a /usr/lib &&
ln -sf libshadow.so.0 libshadow.so &&
cd /usr/lib &&
ln -sf ../../lib/libshadow.so &&
cd /usr/sbin &&
ln -sf vipw vigr &&
cd /usr/share/man/man8 &&
ln -sf vipw.8 vigr.8
```

Command explanations

sed 's/\(.*\) (nflg || \(.*\))\(.*\)/\1\2\3/' src/useradd.c.backup > src/useradd.c &&: This sed is used to fix a compilation bug which occurs due to a variable (nflg) being used but not defined.

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 -sf vipw vigr and **ln -sf vipw.8 vigr.8**: 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−20001016

Program Files

chage, chfn, chpasswd, chsh, dpasswd, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, mkpasswd, newgrp, newusers, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, 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 user full name, office number, office extension, and home phone number information for a user's account.

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

Checks and enforces 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.

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 system authentication information.

pwconv

pwconv converts to shadow passwd files from normal passwd files.

pwunconv

pwunconv converts from shadow passwd files to normal files.

sg

sg executes command as a different group ID.

su

Change the effective user id and group id to that of a user. This replaces the su programs that's installed from the Shellutils package.

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

libshadow.[a,so]

Descriptions

libshadow

libshadow provides common functionality for the shadow programs.

Dependencies

Shadow−20001016 needs the following to be installed:

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 mawk: mawk 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:           1 minute
Estimated required disk space:  710 KB
```

Installation of Sysklogd

Install Sysklogd by running the following commands:

```
make &&
make install
```

Contents of sysklogd−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 a 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.

Dependencies

Sysklogd−1.4.1 needs the following to be installed:

binutils: as, ld, strip fileutils: install gcc: cc1, collect2, cpp0, gcc make: make

Installing Sysvinit−2.84

```
Estimated build time:           1 minute
Estimated required disk space:  630 KB
```

Installation of Sysvinit

When run levels are changed (for example when going to shutdown the system) the init program is going to send the TERM and KILL signals to all the processes that init started. But init prints a message to the screen saying "sending all processes the TERM signal" and the same for the KILL signal. This implies that init sends this signal to all the currently running processes, which isn't the case. To avoid this confusion, you change the init.c file so that the sentence reads "sending all 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 src/init.c.backup &&
sed 's/\(.*\)\(Sending processes\)\(.*\)/\1\2 started by init\3/' \
   src/init.c.backup > src/init.c
```

Install Sysvinit by running the following commands:

```
make −C src &&
make −C src install
```

Contents of sysvinit−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 users terminal by others. It's typically used to allow or disallow other users to write to his terminal.

pidof

pidof finds the process id's (pids) of the named programs and prints those id's on standard output.

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 (e.g., LILO).

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.

Dependencies

Sysvinit−2.84 needs the following to be installed:

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:           1 minute
Estimated required disk space:  7 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 from the toplevel directory:

```
./configure −−prefix=/usr −−libexecdir=/usr/bin \
   −−bindir=/bin &&
make &&
make install
```

Contents of tar−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.

Dependencies

Tar−1.13 needs the following to be installed:

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
mawk: mawk net−tools: hostname patch: patch sed: sed sh−utils: basename, echo, expr, sleep, uname
texinfo: install−info, makeinfo textutils: cat, tr

Installing Textutils−2.0

```
Estimated build time:           1 minute
Estimated required disk space:  15 MB
```

Installation of Textutils

Install Textutils by running the following commands:

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

Contents of textutils−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

Print 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.

Dependencies

Textutils−2.0 needs the following to be installed:

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
mawk: mawk 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−2.11n

```
Estimated build time:           1 minute
Estimated required disk space:  9 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 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−2.11n

Program Files

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize kill, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.minix, mkswap, more, mount, namei, 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, tunelp, ul, umount, vidmode, 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 an 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.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.

kill

kill sends a specified signal to the specified process.

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.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.

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.

Dependencies

Util−linux−2.11n needs the following to be installed:

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−1.6

```
Estimated build time:           1 minute
Estimated required disk space:  6 KB
```

Installation of LFS−Bootscripts

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−1.6

Scripts

checkfs, cleanfs, functions, halt, loadkeys, localnet, mountfs, 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.

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.

network

The network script setup 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.

Dependencies

bootscripts−1.6 needs the following to be installed:

fileutils: chown, cp

Removing old NSS library files

If you have copied the NSS Library files from the normal Linux system to the LFS system (because the normal system runs Glibc−2.0) it's time to remove them now by running:

```
rm /lib/libnss*.so.1 /lib/libnss*2.0*
```

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:

```
cd /etc &&
```

```
ln -sf ../usr/share/zoneinfo/<tzselect's output> 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 localtime
```

Or:

```
ln -sf ../usr/share/zoneinfo/Canada/Eastern localtime
```

Configuring Dynamic Loader

By default the dynamic loader searches a few default paths for dynamic libraries, so there normally isn't a need for the `/etc/ld.so.conf` file unless the system has extra directories in which you want the system to search for libraries. The `/usr/local/lib` directory isn't searched through for dynamic libraries by default, so we want to add this path so when you install software you won't be surprised by them not running for some reason.

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

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

/lib
/usr/lib
/usr/local/lib

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

Although it's not necessary to add the `/lib` and `/usr/lib` directories it doesn't hurt. This way it can be seen right away what's being searched and a you don't have to remember the default search paths if you don't want to.

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 the utilities to modify user's passwords, add new users/groups, delete users/groups and more. We're not going to explain what 'password shadowing' means. All about that can be read in the doc/HOWTO file within the unpacked shadow password suite's source tree. There's one thing you should keep in mind, if you decide to use shadow support, that programs that need to verify passwords (examples are xdm, ftp daemons, pop3 daemons, etc) need to be 'shadow−compliant', e.g. 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:respawn:/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
```

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 setup 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 setup as they deal with hardware dependant information.

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 want to try something else, someone has written an LFS−Hint on BSD style init scripts at http://hints.linuxfromscratch.org/hints/bsd−init.txt .

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 an K, the others begin with an S, and all of them have three 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 PID's.

Feel free to modify the way the boot process works (after all it's your LFS system, not ours). The files here are just an example of how it can be done in a nice way (well what we consider nice anyway. You may hate it).

Configuring the loadkeys script

You only need to use the loadkeys script if you don't have a default 101 keys US keyboard layout.

The /etc/sysconfig/keyboard file contains the information the loadkeys script needs to operate. This file contains the LAYOUT variable which tells loadkeys what keymap to load that corresponds with your keyboard.

Create a new file `/etc/sysconfig/keyboard` by running the following:

```
cat > /etc/sysconfig/keyboard << "EOF"
# Begin /etc/sysconfig/keyboard

LAYOUT=<path-to-keymap>

# End /etc/sysconfig/keyboard
EOF
```

Replace **`<path-to-keymap>`** with the path to the keymap you have selected. For example, if you have chosen the US keymap, you would replace it with **`/usr/share/kbd/keymaps/i386/qwerty/us.map.gz`**

Configuring the setclock script

The setclock script is only for real use when the hardware clock (also known as BIOS or CMOS clock) isn't set to GMT time. The recommended setup is setting the hardware clock to GMT and having the time converted to localtime using the /etc/localtime symbolic link. But if an OS is run that doesn't understand a clock set to GMT (most notable are Microsoft OS'es) you may want to set the clock to localtime so that the time is properly displayed on those OS'es. This script will then set the kernel time to the hardware clock without converting the time using the /etc/localtime symlink.

If you want to use this script on your system even if the hardware clock is set to GMT, then the UTC variable below has to be changed to the value of *1*.

Create a new file `/etc/sysconfig/clock` by running the following:

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

UTC=0

# 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.

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 am existing network).

Configuring the network script

This section only applies if you're going to configure a network card.

Configuring default gateway

If you're on a network you may need to setup 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 boot up 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

# location of filesystem        mount point     fs-type         options

/dev/*LFS-partition device*     /               *fs-type*       defaults 1 1
/dev/*swap-partition device*    swap            swap            pri=1 0 0
proc                            /proc           proc            defaults 0 0

# End /etc/fstab
EOF
```

`*LFS-partition device*`, **`*swap-partition device*`** 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−2.4.17

```
Estimated build time:           Depends on options selected
Estimated required disk space:  Depends on options selected
```

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. 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 (make oldconfig may be better in some situations. See the `README` file for more details when to use make oldconfig).

The following commands are run to build the kernel:

```
cd /usr/src/linux &&
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.

Dependencies

Linux−2.4.17 needs the following to be installed:

bash: sh binutils: ar, as, ld, nm, objcopy fileutils: cp, ln, mkdir, mv, rm findutils: find, xargs
gcc: cc1, collect2, cpp0, gcc grep: grep gzip: gzip make: make mawk: 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, touch

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 <kernel images> $LFS/boot
```

To find out which kernel images files are being used, look at the /etc/lilo.conf file and look for the lines starting with *image=*. If your host system has kernel files in other places than the /boot directory, make sure you update the paths in the $LFS/etc/lilo.conf file so that it does look for them in the /boot directory.

Chapter 9. The End

The End

Well done! You have finished installing your LFS system. It may have been a long process but it was well 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 more often caused by typo's 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 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−3.2−rc2 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. This can just be a null−byte file by running:

```
touch $LFS/etc/lfs-3.2-rc2
```

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
```

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
```

If you are wondering: "Well, where to go now?" you'll be glad to hear that someone has written an LFS hint on the subject at http://hints.linuxfromscratch.org/hints/afterlfs.txt. On a same note, if you are not only newbie to LFS, but also newbie to Linux in general, you may find the newbie hint at http://hints.linuxfromscratch.org/hints/newbie.txt very interesting.

Don't forget there are several LFS mailinglists you can subscribe to if you are in need of help, advice, etc. 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

This appendix describes the following aspects of every package that is installed in this book:

- The official download location for the package.
- What the package contains.
- What each program from a package does.
- What each package needs to compile.

Most information about these packages (especially the descriptions of them) come from the man pages from those packages. We are not going to print the entire man page, just the core elements to make it possible to understand what a program does. To get knowledge of all details on a program, we suggest you start by reading the complete man page in addition to this appendix.

Certain packages are documented in more depth than others, because we just happen to know more about certain packages than I know about others. If 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 current 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. If we upgrade to a major new release, we'll make sure the dependencies are checked too at the same time.

Autoconf

Official Download Location

Autoconf (2.52): ftp://ftp.gnu.org/gnu/autoconf/

Contents of autoconf−2.52

Program Files

autoconf, autoheader, 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

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 source files in the directory tree rooted at a directory given as a command line argument, or the current directory if none is given. It searches the source files for common portability problems and creates a file configure.scan which is a 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 figure out what its configure needs to check for. It may help fill in some gaps in a configure.in generated by autoscan.

Dependencies

Autoconf−2.52 needs the following to be installed:

bash: sh diffutils: cmp fileutils: chmod, install, ln, ls, mkdir, mv, rm grep: fgrep, grep m4: m4 make: make mawk: mawk sed: sed sh−utils: echo, expr, hostname, sleep, uname texinfo: install−info textutils: cat, tr

Automake

Official Download Location

Automake (1.5): ftp://ftp.gnu.org/gnu/automake/

Contents of automake−1.5

Program Files

aclocal and automake

Descriptions

aclocal

Automake includes a number of Autoconf macros which can be used in packages; some of them are actually required by Automake in certain situations. These macros must be defined in the aclocal.m4−file; otherwise 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

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.

Dependencies

Automake−1.5 needs the following to be installed:

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−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. Bash is a program that 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.

Dependencies

Bash−2.05a needs the following to be installed:

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 mawk: awk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info textutils: cat, tr, uniq

Bin86

Official Download Location

Bin86 (0.16.0): http://www.cix.co.uk/~mayday/

Contents of bin86−0.16.0

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

No description is currently available.

objdump86

No description is currently available.

size86

No description is currently available.

Dependencies

Bin86−0.16.0 needs the following to be installed:

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 (2.11.2): ftp://ftp.gnu.org/gnu/binutils/

Contents of binutils−2.11.2

Program Files

addr2line, ar, as, c++filt, 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.

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.

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 a member of an archive 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 objfile 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, libiberty.a and libopcodes.a

Descriptions

libbfd

libbfd is the Binary File Descriptor library.

libiberty

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

libopcodes

No description is currently available.

Dependencies

Binutils−2.11.2 needs the following to be installed:

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 mawk: mawk 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 (1.31): ftp://ftp.gnu.org/gnu/bison/

Contents of bison−1.31

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

Dependencies

Bison−1.31 needs the following to be installed:

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 (1.0.1): ftp://sourceware.cygnus.com/pub/bzip2/

Contents of bzip2−1.0.1

Program Files

bunzip2 (link to bzip2), bzcat (link to bzip2), bzip2 and bzip2recover

Descriptions

bunzip2

Bunzip2 decompresses files that are compressed with bzip2.

bzcat

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

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.

Library Files

libbz2.[a,so]

libbz2

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

Dependencies

Bzip2−1.0.1 needs the following to be installed:

bash: sh binutils: ar, as, ld, ranlib fileutils: cp, ln, rm gcc: cc1, collect2, cpp0, gcc make: make

Chroot

Dependencies

Chroot needs the following to be installed:

bash: bash sh−utils: env

Diffutils

Official Download Location

Diff Utils (2.7): ftp://ftp.gnu.org/gnu/diffutils/

Contents of diffutils−2.7

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.

Dependencies

Diffutils−2.7 needs the following to be installed:

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 (1.25): ftp://download.sourceforge.net/pub/sourceforge/e2fsprogs/
http://download.sourceforge.net/e2fsprogs/

Contents of e2fsprogs−1.25

Program Files

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, 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 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 is used to check and optionally repair Linux second extended filesystems. fsck.ext2 does the same as e2fsck.

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

No description is currently available.

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.

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 among UUIDs created 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

No description is currently available.

libe2p

No description is currently available.

libext2fs

No description is currently available.

libss

No description is currently available.

libuuid

No description is currently available.

Dependencies

E2fsprogs−1.25 needs the following to be installed:

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 mawk: 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/

Contents of ed−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.

Dependencies

Ed−0.2 needs the following to be installed:

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 (3.37): ftp://ftp.gw.com/mirrors/pub/unix/file/

Contents of file−3.37

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.

Dependencies

File−3.37 needs the following to be installed:

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 mawk: mawk sed: sed sh−utils: echo, expr, hostname, sleep texinfo: makeinfo
textutils: cat, tr

Fileutils

Official Download Location

File Utils (4.1): ftp://ftp.gnu.org/gnu/fileutils/

Contents of fileutils−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 argument and for each subdirectory of directory arguments.

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.

Dependencies

Fileutils−4.1 needs the following to be installed:

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

Find Utils (4.1): ftp://ftp.gnu.org/gnu/findutils/ Find Utils Patch (4.1): ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of findutils−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

updatedb runs a program called frcode 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 contain 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 system 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 file can be created that contains all these files (one per line) and use xargs to perform that command on the list.

Dependencies

Findutils−4.1 needs the following to be installed:

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−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 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 sets up rules for what to look for than to write the actual program that finds the text.

flex++

flex++ invokes a version of flex which is used exclusively for C++ scanners.

lex

We create a yacc script 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

No description is currently available.

Dependencies

Flex−2.5.4a needs the following to be installed:

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

GCC

Official Download Location

GCC (2.95.3): ftp://ftp.gnu.org/pub/gnu/gcc/ GCC Patch (2.95.3−2): ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of gcc−2.95.3

Program Files

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

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

c++filt is used to demangle C++ symbols.

collect2

No description is currently available.

cpp, cpp0

cpp pre−processes a source file, such as including the contents of header files into the source file. It's a good idea to not do this manually to save a lot of time. Someone just inserts a line like #include <filename>. The preprocessor inserts the contents of that file into the source file. That's one of the things a preprocessor does.

gcov

No description is currently available.

protoize

No description is currently available.

unprotoize

No description is currently available.

Library Files

libgcc.a, libiberty.a, libstdc++.[a,so]

libgcc

libgcc.a is a run−time support file for gcc.Most of the time, on most machines, libgcc.a is not actually necessary.

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.

Dependencies

GCC−2.95.3 needs the following to be installed:

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 (0.10.40): ftp://ftp.gnu.org/gnu/gettext/

Contents of gettext−0.10.40

Program Files

gettext, gettextize, msgcmp, msgcomm, msgfmt, msgmerge, msgunfmt, ngettext and xgettext

Descriptions

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 users native language rather than in the default English language.

gettextize

No description is currently available.

msgcmp

No description is currently available.

msgcomm

No description is currently available.

msgfmt

No description is currently available.

msgmerge

No description is currently available.

msgunfmt

No description is currently available.

ngettext

No description is currently available.

xgettext

No description is currently available.

Dependencies

Gettext−0.10.40 needs the following to be installed:

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 mawk: mawk 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/

Contents of glibc−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

No description is currently available.

gencat

gencat generates message catalogues.

getconf

No description is currently available.

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

No description is currently available.

locale

No description is currently available.

localedef

localedef compiles locale specifications.

mtrace

No description is currently available.

nscd

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

nscd_nischeck

No description is currently available.

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

No description is currently available.

rpcinfo

No description is currently available.

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

No description is currently available.

libSegFault

No description is currently available.

libanl, libanl_p

No description is currently available.

libbsd−compat

No description is currently available.

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) come in two flavors: dynamic ones and static ones. In short when a program uses a static C library, the code from the C library will be copied into the executable file. When a program uses a dynamic library, that 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

No description is currently available.

libg

No description is currently available.

libieee

No description is currently available.

libm, libm_p

libm is the mathematical library.

libmcheck

No description is currently available.

libmemusage

No description is currently available.

libnsl, libnsl_p

No description is currently available.

libnss_compat, libnss_dns, libnss_files, libnss_hesiod, libnss_nis, libnss_nisplus

No description is currently available.

libpcprofile

No description is currently available.

libpthread, libpthread_p

No description is currently available.

libresolv, libresolv_p

No description is currently available.

librpcsvc, librpcsvc_p

No description is currently available.

librt, librt_p

No description is currently available.

libthread_db

No description is currently available.

libutil, libutil

No description is currently available.

Dependencies

Glibc−2.2.5 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: date, expr, hostname, pwd, uname texinfo: install−info, makeinfo textutils: cat, cut, sort, tr

Grep

Official Download Location

Grep (2.4.2): ftp://ftp.gnu.org/gnu/grep/

Contents of grep–2.4.2

Program Files

egrep, fgrep 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.

Dependencies

Grep–2.4.2 needs the following to be installed:

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
mawk: mawk sed: sed sh–utils: basename, echo, expr, hostname, sleep, uname
texinfo: install–info, makeinfo textutils: cat, tr

Groff

Official Download Location

Groff (1.17.2): ftp://ftp.gnu.org/gnu/groff/

Contents of groff−1.17.2

Program Files

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

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.

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.

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.

Dependencies

Groff−1.17.2 needs the following to be installed:

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 mawk: 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/ Gzip Patch (1.2.4a):
ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of gzip−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 either a list of files on the command line or its standard input and writes the uncompressed data on standard output

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.

Dependencies

Gzip−1.2.4a needs the following to be installed:

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 (1.06−2): ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/ http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of kbd−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 it is by using this program that 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.

Dependencies

Kbd−1.06 needs the following to be installed:

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

Linux kernel

Official Download Location

Linux Kernel (2.4.17): ftp://ftp.kernel.org/pub/linux/kernel/

Contents of kernel−2.4.17

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 such as 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 so should *not* be replaced when upgrading the kernel. They are essential for compiling many programs.

Dependencies

Linux−2.4.17 needs the following to be installed:

bash: sh binutils: ar, as, ld, nm, objcopy fileutils: cp, ln, mkdir, mv, rm findutils: find, xargs
gcc: cc1, collect2, cpp0, gcc grep: grep gzip: gzip make: make mawk: 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, touch

Less

Official Download Location

Less (358): ftp://ftp.gnu.org/gnu/less/

Contents of less−358

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 with 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.

Dependencies

Less−358 needs the following to be installed:

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

Libtool

Official Download Location

Libtool (1.4.2): ftp://ftp.gnu.org/gnu/libtool/

Contents of libtool−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,so]

Descriptions

libltdl

Libtool provides a small library, called `libltdl', that aims at hiding the various difficulties of dlopening libraries from programmers.

Dependencies

Libtool−1.4.2 needs the following to be installed:

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 (22.1): ftp://ibiblio.org/pub/Linux/system/boot/lilo/ http://ibiblio.org/pub/Linux/system/boot/lilo/

Contents of lilo−22.1

Program Files

lilo and mkrescue

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.

Dependencies

Lilo−22.1 needs the following to be installed:

bash: sh bin86: as86, ld86 binutils: as, ld, strip fileutils: cp, dd, ln gcc: cc, cc1, collect2, cpp0
make: make sed: sed textutils: cat

M4

Official Download Location

M4 (1.4): ftp://ftp.gnu.org/gnu/m4/

Contents of m4−1.4

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. M4 can be used either as a front−end to a compiler or as a macro processor in its own right.

Dependencies

M4−1.4 needs the following to be installed:

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−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.

Dependencies

Make−3.79.1 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: install−info, makeinfo
textutils: cat, tr

MAKEDEV

Official Download Location

MAKEDEV (1.4): ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of MAKEDEV−1.4

Program Files

MAKEDEV

Descriptions

MAKEDEV

MAKEDEV is a script that can help in creating the necessary static device files that usually reside in the /dev directory. More information on device nodes can be found in the Linux Kernel source tree in `Documentation/devices.txt`.

Dependencies

MAKEDEV−1.4 needs the following to be installed:

bash: sh fileutils: chmod, chown, cp, ln, mknod, mv, rm grep: grep sh−utils: expr, id

Man

Official Download Location

Man (1.5j): ftp://ftp.win.tue.nl/pub/linux−local/utils/man/

Contents of man−1.5j

Program Files

apropos, makewhatis, man, man2dvi, man2html and whatis

Descriptions

apropos

apropos searches a set of database files containing short descriptions of system commands for keywords 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 a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output. Only complete word matches are displayed.

Dependencies

Man−1.5i2 needs the following to be installed:

bash: sh binutils: as, ld fileutils: chmod, cp, install, mkdir, rm gcc: c11, collect2, cpp0, gcc grep: grep make: make mawk: awk sed: sed sh−utils: echo textutils: cat

Man−pages

Official Download Location

Man−pages (1.47): ftp://ftp.kernel.org/pub/linux/docs/manpages/

Contents of manpages−1.47

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.

Dependencies

Man−pages−1.47 needs the following to be installed:

bash: sh fileutils: install make: make

Mawk

Official Download Location

Mawk (1.3.3): ftp://ftp.whidbey.net/pub/brennan/

Contents of mawk−1.3.3

Program Files

awk (link to mawk) and mawk

Descriptions

awk

awk is symlinked to mawk for programs which just look for any generic awk.

mawk

Mawk is an interpreter for the AWK Programming Language. The AWK language is useful for manipulation of data files, text retrieval and processing, and for prototyping and experimenting with algorithms.

Dependencies

Mawk−1.3.3 needs the following to be installed:

bash: sh fileutils: chmod, cp, ln, rm binutils: as, ld diffutils: cmp gcc: cc1, collect2, cpp0, gcc
grep: egrep, grep make: make sed: sed sh−utils: hostname, tee textutils: cat, tr

Modutils

Official Download Location

Modutils (2.4.12): ftp://ftp.kernel.org/pub/linux/utils/kernel/modutils/

Contents of modutils–2.4.12

Program Files

depmod, genksyms, insmod, insmod_ksymoops_clean, kallsyms (link to insmod), kernelversion, ksyms, lsmod (link to insmod), modinfo, modprobe (link to insmod) and rmmod

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.

Dependencies

Modutils−2.4.12 needs the following to be installed:

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/

Contents

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

The libraries that make up the Ncurses library 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. The libncurses libraries are the base of the system.

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.

Dependencies

Ncurses−5.2 needs the following to be installed:

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
mawk: mawk 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−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.

Dependencies

Netkit−base−0.17 needs the following to be installed:

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−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 is used to set or show the system's hostname

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

updomainname shows or sets the system's NIS/YP domain name.

Dependencies

Net−tools−1.60 needs the following to be installed:

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−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 1MB 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 1MB or just as a patch file of 1KB 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.

Dependencies

Patch−2.5.4 needs the following to be installed:

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 (5.6.1): http://www.perl.com/

Contents of perl–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

pos2usage 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.

Dependencies

Perl−5.6.1 needs the following to be installed:

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 mawk: 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−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.

Dependencies

Procinfo−18 needs the following to be installed:

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/

Contents of procps−2.0.7

Program Files

free, kill, oldps, pgrep, pkill, ps, skill, snice, sysctl, tload, top, uptime, 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.

uptime

uptime gives a one line display of the following information: the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

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.

Dependencies

Procps−2.0.7 needs the following to be installed:

bash: sh binutils: as, ld, strip fileutils: install, ln, mv, rm gcc: cc1, collect2, cpp0, gcc grep: grep make: make mawk: awk sed: sed sh−utils: basename, pwd textutils: sort, tr

Psmisc

Official Download Location

Psmisc (20.2): http://download.sourceforge.net/psmisc/
ftp://download.sourceforge.net/pub/sourceforge/psmisc/

Contents of psmisc−20.2

Program Files

fuser, killall, pidof (link to 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 using the specified files or file systems.

killall

killall sends a signal to all processes running any of the specified commands.

pidof

Pidof finds the process id's (pids) of the named programs and prints those id's on standard output.

pstree

pstree shows running processes as a tree.

Dependencies

Psmisc−20.2 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: basename, echo, expr, hostname, sleep, uname texinfo: makeinfo textutils: cat, tr

Reiserfsprogs

Official Download Location

Reiserfs (3.x.0j): ftp://ftp.namesys.com/pub/reiserfsprogs/

Contents of reiserfsprogs−3.x.0j

Program Files

debugreiserfs, mkreiserfs, reiserfsck, resize_reiserfs and unpack

Descriptions

debugreiserfs

debugreiserfs can sometimes help to solve problems with reiserfs filesystems. If it is called without options it prints the super block of any reiserfs filesystem found on the device.

mkreiserfs

mkreiserfs creates a reiserfs file system.

reiserfsck

reiserfsck checks a reiserfs file system.

resize_reiserfs

resize_reiserfs is used to resize an unmounted reiserfs file system

unpack

No description is currently available.

Dependencies

Reiserfs−3.x.0j needs the following to be installed:

autoconf: autoconf, autoheader automake: aclocal, automake bash: sh binutils: ar, as, ld, ranlib diffutils: cmp fileutils: chmod, install, ls, rm gcc: cc1, collect2, cpp0, gcc grep: egrep, grep m4: m4 make: make mawk: mawk sed: sed sh−utils: echo, expr, hostname, sleep texinfo: makeinfo textutils: cat, tr

Sed

Official Download Location

Sed (3.02): ftp://ftp.gnu.org/gnu/sed/

Contents of sed−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).

Dependencies

Sed−3.02 needs the following to be installed:

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 mawk: mawk sed: sed
sh−utils: echo, expr, hostname, sleep texinfo: install−info, makeinfo textutils: cat, tr

Shadow Password Suite

Official Download Location

Shadow Password Suite (20001016): ftp://ftp.pld.org.pl/software/shadow/

Contents of shadow−20001016

Program Files

chage, chfn, chpasswd, chsh, dpasswd, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, mkpasswd, newgrp, newusers, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, 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 user full name, office number, office extension, and home phone number information for a user's account.

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

Checks and enforces 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.

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 system authentication information.

pwconv

pwconv converts to shadow passwd files from normal passwd files.

pwunconv

pwunconv converts from shadow passwd files to normal files.

sg

sg executes command as a different group ID.

su

Change the effective user id and group id to that of a user. This replaces the su programs that's installed from the Shellutils package.

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

libshadow.[a,so]

Descriptions

libshadow

libshadow provides common functionality for the shadow programs.

Dependencies

Shadow−20001016 needs the following to be installed:

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 mawk: mawk 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/3.2−rc2/ http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of sh−utils−2.0

Program Files

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, hostname, 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 the groups a user is in.

hostid

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

hostname

hostname sets or prints the name of the current host system

id

id prints the real and effective UIDs and GIDs of a user or the current 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 non–tty

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 printf C 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's effective userid.

yes

yes outputs a string repeatedly until killed.

Dependencies

Sh−utils−2.0 needs the following to be installed:

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 mawk: mawk 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): ftp://ibiblio.org/pub/Linux/system/daemons/ http://ibiblio.org/pub/Linux/system/daemons/

Contents of sysklogd−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 a 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.

Dependencies

Sysklogd−1.4.1 needs the following to be installed:

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−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 users terminal by others. It's typically used to allow or disallow other users to write to his terminal.

pidof

pidof finds the process id's (pids) of the named programs and prints those id's on standard output.

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 (e.g., LILO).

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.

Dependencies

Sysvinit−2.84 needs the following to be installed:

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/3.2−rc2/ http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of tar−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.

Dependencies

Tar−1.13 needs the following to be installed:

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
mawk: mawk 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 (4.0): ftp://ftp.gnu.org/gnu/texinfo/

Contents of texinfo−4.0

Program Files

info, 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 be more in depth than just explaining the options to a program.

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, it is also necessary to delete the topic in the index file as well. 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.

Dependencies

Texinfo−4.0 needs the following to be installed:

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

Text Utils (2.0): ftp://ftp.gnu.org/gnu/textutils/

Contents of textutils–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

Print 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.

Dependencies

Textutils−2.0 needs the following to be installed:

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
mawk: mawk 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 (2.11n): ftp://ftp.win.tue.nl/pub/linux–local/utils/util–linux/

Contents of util–linux–2.11n

Program Files

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize kill, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.minix, mkswap, more, mount, namei, 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, tunelp, ul, umount, vidmode, 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 an 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.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.

kill

kill sends a specified signal to the specified process.

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.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.

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.

Dependencies

Util−linux−2.11n needs the following to be installed:

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 (6.0): ftp://ftp.vim.org/pub/editors/vim/unix/

Contents

Program Files

ex (link to vim), rview (link to vim), rvim (link to vim), vi (link to vim), view (link to vim), vim, vimdiff (link to vim), vimtutor (link to vim) and xxd

Descriptions

ex

ex starts vim in Ex mode.

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.

vi

vi starst vim in vi–compatible mode.

view

view starts vim in read–only mode.

vim

vim starts vim in the normal, default way.

vimdiff

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

vimtutor

vimtutor starts the Vim tutor.

xxd

xxd makes a hexdump or does the reverse.

Dependencies

Vim−6.0 needs the following to be installed:

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

LFS−Bootscripts

Official Download Location

LFS−Bootscripts (1.6): ftp://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/
http://ftp.linuxfromscratch.org/lfs−packages/3.2−rc2/

Contents of LFS−bootscripts−1.6

Scripts

checkfs, cleanfs, functions, halt, loadkeys, localnet, mountfs, 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.

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.

network

The network script setup 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.

Dependencies

bootscripts−1.6 needs the following to be installed:

fileutils: chown, cp

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.linuxdoc.org

- Linux Network Administrator's Guide
- From−PowerUp−To−Bash−Prompt−HOWTO

Other

- The various man and info pages that come with the packages