Linux From Scratch

Version 6.4

Gerard Beekmans

Linux From Scratch: Version 6.4

by Gerard Beekmans

Table of Contents

Preface

Foreword

My adventures in Linux began in 1998 when I downloaded and installed my first distribution. After working with it for a while, I discovered issues I definitely would have liked to see improved upon. For example, I didn't like the arrangement of the bootscripts or the way programs were configured by default. I tried a number of alternative distributions to address these issues, yet each had its pros and cons. Finally, I realized that if I wanted full satisfaction from my Linux system, I would have to build my own from scratch.

What does this mean? I resolved not to use pre-compiled packages of any kind, nor CD-ROMs or boot disks that would install basic utilities. I would use my current Linux system to develop my own customized system. This “perfect” Linux system would then have the strengths of various systems without their associated weaknesses. In the beginning, the idea was rather daunting, but I remained committed to the idea that a system could be built that would conform to my needs and desires rather than to a standard that just did not fit what I was looking for.

After sorting through issues such as circular dependencies and compile-time errors, I created a custom-built Linux system that was fully operational and suitable to individual needs. This process also allowed me to create compact and streamlined Linux systems which are faster and take up less space than traditional operating systems. I called this system a Linux From Scratch system, or an LFS system for short.

As I shared my goals and experiences with other members of the Linux community, it became apparent that there was sustained interest in the ideas set forth in my Linux adventures. Such custom-built LFS systems serve not only to meet user specifications and requirements, but also serve as an ideal learning opportunity for programmers and system administrators to enhance their Linux skills. Out of this broadened interest, the Linux From Scratch Project was born.

This *Linux From Scratch* book provides readers with the background and instruction to design and build custom Linux systems. This book highlights the Linux from Scratch project and the benefits of using this system. Users can dictate all aspects of their system, including directory layout, script setup, and security. The resulting system will be compiled completely from the source code, and the user will be able to specify where, why, and how programs are installed. This book allows readers to fully customize Linux systems to their own needs and allows users more control over their system.

I hope you will have a great time working on your own LFS system, and enjoy the numerous benefits of having a system that is truly *your own*.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Audience

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

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

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

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

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

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

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

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

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

Prerequisites

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

Because the LFS book assumes *at least* this basic level of skill, the various LFS support forums are unlikely to be able to provide you with much assistance; you will find that your questions regarding such basic knowledge will likely go unanswered, or you will simply be referred to the LFS essential pre-reading list.

Before building an LFS system, we recommend reading the following HOWTOs:

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

  This is a comprehensive guide to building and installing “generic” Unix software packages under Linux.
- The Linux Users' Guide *http://www.linuxhq.com/guides/LUG/guide.html*

  This guide covers the usage of assorted Linux software.
- The Essential Pre-Reading Hint *http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt*

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

Host System Requirements

Your host system should have the following software with the minimum versions indicated. This should not be an issue for most modern Linux distributions. Also note that many distributions will place software headers into separate packages, often in the form of “<package-name>-devel” or “<package-name>-dev”. Be sure to install those if your distribution provides them.

- **Bash-2.05a** (/bin/sh should be a symbolic or hard link to bash)
- **Binutils-2.12** (Versions greater than 2.18 are not recommended as they have not been tested)
- **Bison-1.875** (/usr/bin/yacc should be a link to bison or small script that executes bison)
- **Bzip2-1.0.2**
- **Coreutils-5.0** (or Sh-Utils-2.0, Textutils-2.0, and Fileutils-4.1)
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.0** (/usr/bin/awk should be a link to gawk)
- **Gcc-3.0.1** (Versions greater than 4.3.2 are not recommended as they have not been tested)
- **Glibc-2.2.5** (Versions greater than 2.8-20080929 are not recommended as they have not been tested)
- **Grep-2.5**
- **Gzip-1.2.4**
- **Linux Kernel-2.6.x** (having been compiled with GCC-3.0 or greater)

  The reason for the kernel version requirement is that thread-local storage support in Binutils will not be built and the Native POSIX Threading Library (NPTL) test suite will segfault if the host's kernel isn't at least a 2.6.x version compiled with a 3.0 or later release of GCC.

  If the host kernel is either earlier than 2.6.x, or it was not compiled using a GCC-3.0 (or later) compiler, you will have to replace the kernel with one adhering to the specifications. There are two methods you can take to solve this. First, see if your Linux vendor provides a 2.6 kernel package. If so, you may wish to install it. If your vendor doesn't offer a 2.6 kernel package, or you would prefer not to install it, then you can compile a 2.6 kernel yourself. Instructions for compiling the kernel and configuring the boot loader (assuming the host uses GRUB) are located in Chapter 8.

Note

This version of the book builds a 32-bit Linux system and requires an existing 32-bit version of of the kernel on the Intel/AMD x86 architecture. Adding capabilty for x86_64 systems is a major objective of a future version of LFS. Support for 64-bit systems and additional architectures can be found in the Cross-Compiled Linux From Scratch (CLFS) project at *http://cross-lfs.org/view/svn/*.

- **M4-1.4**
- **Make-3.79.1**
- **Patch-2.5.4**
- **Perl-5.6.0**
- **Sed-3.0.2**
- **Tar-1.14**
- **Texinfo-4.8**

Note that the symlinks mentioned above are required to build an LFS system using the instructions contained within this book. Symlinks that point to other software (such as dash, mawk, etc.) may work, but are not tested or supported by the LFS development team, and may require either deviation from the instructions or additional patches to some packages.

To see whether your host system has all the appropriate versions, and the ability to compile programs, run the following:

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

# Simple script to list version numbers of critical development tools

bash --version | head -n1 | cut -d" " -f2-4
echo "/bin/sh -> `readlink -f /bin/sh`"
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
if [ -e /usr/bin/yacc ];
  then echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
  else echo "yacc not found"; fi
bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
if [ -e /usr/bin/awk ];
  then echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
  else echo "awk not found"; fi
gcc --version | head -n1
/lib/libc.so.6 | head -n1 | cut -d" " -f1-7
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1
echo 'main(){}' > dummy.c && gcc -o dummy dummy.c
if [ -x dummy ]; then echo "Compilation OK";
  else echo "Compilation failed"; fi
rm -f dummy.c dummy

EOF

bash version-check.sh
```

Typography

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

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

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

In some cases, a logical line is extended to two or more physical lines with a backslash at the end of the line.

```
CC="gcc -B/usr/bin/" ../binutils-2.18/configure \
  --prefix=/tools --disable-nls --disable-werror
```

Note that the backslash must be followed by an immediate return. Other whitespace characters like spaces or tab characters will create incorrect results.

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

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

Emphasis

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

http://www.linuxfromscratch.org/

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

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

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

<REPLACED TEXT>

This format is used to encapsulate text that is not to be typed as seen or for copy-and-paste operations.

[OPTIONAL TEXT]

This format is used to encapsulate text that is optional.

`passwd(5)`

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

Structure

This book is divided into the following parts.

Part I - Introduction

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

Part II - Preparing for the Build

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

Part III - Building the LFS System

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

Errata

The software used to create an LFS system is constantly being updated and enhanced. Security warnings and bug fixes may become available after the LFS book has been released. To check whether the package versions or instructions in this release of LFS need any modifications to accommodate security vulnerabilities or other bug fixes, please visit *http://www.linuxfromscratch.org/lfs/errata/6.4/* before proceeding with your build. You should note any changes shown and apply them to the relevant section of the book as you progress with building the LFS system.

Part I. Introduction

Chapter 1. Introduction

1.1. How to Build an LFS System

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

As an alternative to installing a separate distribution onto your machine, you may wish to use the Linux From Scratch LiveCD or a LiveCD from a commercial distribution. The LFS LiveCD works well as a host system, providing all the tools you need to successfully follow the instructions in this book. Unfortunately, development of the LiveCD has not progressed recently and it only contains older versions of the source packages and patches (for those ISOs not labels -nosrc or -min), and this book. For more information about the LFS LiveCD or to download a copy, visit *http://www.linuxfromscratch.org/livecd/*.

> **Note**
>
> The LFS LiveCD might not work on newer hardware configurations, failing to boot or failing to detect some devices such as some SATA hard drives.

Chapter 2 of this book describes how to create a new Linux native partition and file system, the place where the new LFS system will be compiled and installed. Chapter 3 explains which packages and patches need to be downloaded to build an LFS system and how to store them on the new file system. Chapter 4 discusses the setup of an appropriate working environment. Please read Chapter 4 carefully as it explains several important issues the reader should be aware of before beginning to work through Chapter 5 and beyond.

Chapter 5 explains the installation of a number of packages that will form the basic development suite (or toolchain) which is used to build the actual system in Chapter 6. Some of these packages are needed to resolve circular dependencies—for example, to compile a compiler, you need a compiler.

Chapter 5 also shows the user how to build a first pass of the toolchain, including Binutils and GCC (first pass basically means these two core packages will be reinstalled). The next step is to build Glibc, the C library. Glibc will be compiled by the toolchain programs built in the first pass. Then, a second pass of the toolchain will be built. This time, the toolchain will be dynamically linked against the newly built Glibc. The remaining Chapter 5 packages are built using this second pass toolchain. When this is done, the LFS installation process will no longer depend on the host distribution, with the exception of the running kernel.

This effort to isolate the new system from the host distribution may seem excessive, but a full technical explanation is provided in Section 5.2, “Toolchain Technical Notes”.

In Chapter 6, the full LFS system is built. The **chroot** (change root) program is used to enter a virtual environment and start a new shell whose root directory will be set to the LFS partition. This is very similar to rebooting and instructing the kernel to mount the LFS partition as the root partition. The system does not actually reboot, but instead **chroot**'s because creating a bootable system requires additional work which is not necessary just yet. The major advantage is that “chrooting” allows the builder to continue using the host while LFS is being built. While waiting for package compilation to complete, a user can switch to a different virtual console (VC) or X desktop and continue using the computer as normal.

To finish the installation, the LFS-Bootscripts are set up in Chapter 7, and the kernel and boot loader are set up in Chapter 8. Chapter 9 contains information on furthering the LFS experience beyond this book. After the steps in this book have been implemented, the computer will be ready to reboot into the new LFS system.

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

1.2. What's new since the last release

Below is a list of package updates made since the previous release of the book.

Upgraded to:

- Autoconf 2.63
- Automake 1.10.1
- Berkeley DB 4.7.25
- Binutils 2.18
- Bzip2 1.0.5
- Coreutils 6.12
- E2fsprogs 1.41.3
- File 4.26
- Findutils 4.4.0
- Flex 2.5.35
- Gawk 3.1.6
- GCC 4.3.2
- Gettext 0.17
- Glibc 2.8-20080929
- Grep 2.5.3
- IANA-Etc 2.30
- IPRoute2 2.6.26
- Kbd 1.14.1
- Less 418
- LFS-Bootscripts 20081031
- Libtool 2.2.6a
- Linux 2.6.27.4
- M4 1.4.12
- Man-DB 2.5.2
- Man-pages 3.11
- Module-Init-Tools 3.4.1
- Perl 5.10.0
- Psmisc 22.6
- Shadow 4.1.2.1
- Sysklogd 1.5

- Tar 1.20
- TCL 8.5.5
- Texinfo 4.13a
- Udev 130
- udev-config-20081015
- Util-Linux-NG 2.14.1
- Vim 7.2

Added:

- bash-3.2-fixes-8.patch
- binutils-2.18-configure-1.patch
- binutils-2.18-GCC43-1.patch
- coreutils-6.12-old_build_kernel-1.patch
- coreutils-6.12-i18n-2.patch
- db-4.7.25-upstream_fixes-1.patch
- expect-5.43.0-tcl_8.5.5_fix-1.patch
- GMP-4.2.4
- glibc-2.8-20080929-iconv_tests-1.patch
- glibc-2.8-20080929-ildoubl_test-1.patch
- grep-2.5.3-debian_fixes-1.patch
- grep-2.5.3-upstream_fixes-1.patch
- grub-0.97-256byte_inode-1.patch
- M4 to the Chapter 5 build
- module-init-tools-3.4.1-manpages-1.patch
- MPFR-2.3.2
- perl-5.10.0-consolidated-1.patch
- procps-3.2.7-watch_unicode-1.patch
- readline-5.2-fixes-5.patch
- vim-7.2-fixes-3.patch

Removed:

- bash-3.2-fixes-5.patch
- coreutils-6.10-i18n-1.patch
- db-4.5.20-fixes-1.patch
- gawk-3.1.5-segfault_fix-1.patch
- gcc-4.1.2-specs-1.patch
- grep-2.5.1-redhat_fixes-2.patch
- kbd-1.12-gcc4_fixes-1.patch
- man-db-2.4.4-fixes-1.patch

- mktemp 1.5
- module-init-tools-3.2.2-modprobe-1.patch
- perl-5.8.8-libc-2.patch
- readline-5.2-fixes-3.patch
- shadow-4.0.18.1-useradd_fix-2.patch
- sysklogd-1.4.1-8bit-1.patch
- sysklogd-1.4.1-fixes-2.patch
- Util-linux 2.12r
- vim-7.1-fixes-6.patch

1.3. Changelog

This is version 6.4 of the Linux From Scratch book, dated November 23, 2008. If this book is more than six months old, a newer and better version is probably already available. To find out, please check one of the mirrors via *http://www.linuxfromscratch.org/mirrors.html*.

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

Changelog Entries:

- 2008-11-23
  - [bdubbs] - Released LFS-6.4.
- 2008-11-05
  - [bdubbs] - Changed wording introducing test suites in Chapter 5.
- 2008-10-31
  - [dj] - Updated to lfs-bootscripts-20081031.
- 2008-10-30
  - [bdubbs] - Added explanation for --disable-libssp to GCC in Chapter 5. Also expanded/added explanation on language selection for GCC in Chapters 5 and 6.
  - [bdubbs] - Wording changes to several text sections of Chapter 5. Thanks to Chris Staub for the patch.
  - [bdubbs] - Added a consolidated patch to perl to address security and other issues. Changed the configure options for perl to define a vendor library location.
- 2008-10-29
  - [bdubbs] - Updated symbolic link creation loop for vi.1 man pages. Thanks to Bryan Kadzban for the construct.
- 2008-10-28
  - [bdubbs] - Updated to Tcl-8.5.5.
  - [bdubbs] - Updated to latest stable kernel 2.6.27.4.
  - [bdubbs] - Changed location of man pages in Module-Init-Tools. Thanks to Trent Shea for pointing out the fix.
  - [bdubbs] - Updated to M4-1.4.12.

- 2008-10-27
  - [bdubbs] - Added chmod instructions to e2fsprogs and tcl to ensure all libraries are writable by root for stripping.
  - [bdubbs] - Added a brief explanation of the Linux API Headers instructions.
  - [bdubbs] - Added i386, linux32, and linux64 as symbolic links to setarch in util-linux contents.
  - [bdubbs] - Moved gawk ahead of findutils in Chapter 6 to avoid a test suite failure in findutils.
- 2008-10-26
  - [bdubbs] - Added a General Compilation Instructions section immediatley before binutils. Essentially reordered the presentation that was in the Chapter 5 Introduction.
  - [bdubbs] - Remove unnecessary mandir patch. Updated vim package contents.
- 2008-10-25
  - [dj] - Updated the text on the Man-DB page to accout for recent changes in Man-DB. Thanks to Alexander Patrakov for providing most of the included text, explanations, and examples.
- 2008-10-23
  - [dj] - Updated to lfs-bootscritps-20081023 to account for changes in the console page.
  - [dj] - Updated text in console page to match current situation regarding linux kernel changes. Thanks to Alexander Patrakov for the text and explanations.
  - [dj] - Updated Man-DB instructions and text covering manual pages and related i18n issues.
- 2008-10-22
  - [dj] - Corrected chown command for coreutils testsuite.
  - [dj] - Updated to coreutils-6.12-i18n-2.patch. Thanks to Bryan Kadzban for the suggested fix.
- 2008-10-21
  - [matthew] - Added dependency information for GMP and MPFR packages. Thanks to Chris Staub for the patch. Also, removed dependency information for Mktemp. Thanks to William Immendorf for the report. Fixes #2218.
  - [dj] - Updated list of minimum installed locales for testsuite coverage in Chapter 6 GLibc instructions.
  - [bdubbs] - Added ac_cv_func_working_mktime=yes to the configure commands in gawk and bash to bypass the search for mktime. This works around a change in gcc.
  - [bdubbs] - Added a note to the ifcfg script description in iproute2 that it requires external programs.
  - [dj] - Added '--without-included-regex' to grep instrcutions in order to force the use of glibc's regex library. This fixes the -i switch for grep.
  - [dj] - Reintroduced the command to supress installation of the vi_VN.TCVN locale as bash is still broken with it.
  - [dj] - Put Coreutils-i18n patch back into place.
- 2008-10-20
  - [jhuntwork] - GCC 4.3.2 uses a new directory for fixed includes. Fixed the adjust toolchain scripts to point to the new location.
- 2008-10-19

- [bdubbs] - Added a note to the Host System Requirements that the Linux host must be a 32-bit system and that the book only supports a 32-bit build.
- [randy] - Updated the book to use 4.13a as the Texinfo version, even though the tarball is exactly the same as the previous 4.13 version.
- [randy] - Removed an unnecessary command from the Chapter 5 Perl instructions.
- [bdubbs] - Updated the discussion in Chapter 1 explaining that the LiveCD is out of date.
- [bdubbs] - Added a paragraph to the note in the packages page explaining that bandwidth can be saved when making multiple updates within a minor kernel release by downloading a base version and patches.

- 2008-10-18
  - [jhuntwork] - Fixed build locations of m4 so that it links against the glibc built in /tools and so that no packages in chapter 6 hard-code references to the temporary location. Also made m4 a host requirement.
- 2008-10-15
  - [bdubbs] - Added --disable-libssp to glibc Pass 1 in Chapter 5 to eliminate a build failure on some systems.
  - [dj] - Updated to udev-config-20081015.
  - [dj] - Modified udev instructions following upstream recommendations.
- 2008-10-13
  - [randy] - Modified the Chapter 5 instructions so that instead of building the GMP and MPFR packages separately for GCC Pass2, they are built by GCC internally.
  - [randy] - Added a configure option to the Chapter 6 Gettext instructions so that the documentation is installed in a versioned directory.
- 2008-10-12
  - [dj] - Updated to E2fsprogs-1.41.2.
  - [dj] - Corrected installation prefixes of Iproute2 package with DESTDIR and MANDIR paths. Thanks to Steffen Pankratz for the fix.
  - [randy] - Modified the Chapter 6 GMP instructions to include a method for determining all the tests in the test suite passed.
  - [randy] - Modified the GCC search for correct headers command to account for the new include-fixed directory.
  - [randy] - Added a patch to the Chapter 6 Binutils instructions to correct some errors in the test suite.
  - [dj] - Corrected installation of udev rule files.
  - [randy] - Moved the Chapter 6 M4 installation into alphabetical order as it is installed in Chapter 5 now and therefore doesn't need to precede the Bison installation.
  - [randy] - Moved the Chapter 5 M4 installation to before GCC Pass1 so that the internal GCC build of GMP will not fail in case M4 doesn't exist on the host. Also updated GCC's dependencies to reflect GMP and MPFR.
  - [dj] - Changed Chapter 5 GCC Pass 1 build to static. Thanks to Jeremy Huntwork for the suggestion and supporting text.
  - [dj] - Added note to Chapter 6 GCC about the new include-fixed directory and changed the sample output to match.

  - [dj] - Added instruction to keep Chapter 5 Glibc from honoring /etc/ld.so.preload. Thanks to Alexander Patrakov for the fix.
  - [randy] - Added descriptions of the configure options used in the GMP instructions and updated the installed library descriptions.
- 2008-10-11
  - [dj] - Removed the Chapter 5 Glibc test suite information as it requires a working C++ compiler to run.
  - [randy] - Added three configure parameters to the Chapter 6 Util-linux-ng instructions so that additional programs are installed. Also updated the installed programs list.
  - [randy] - Added a sed command to the Sysvinit instructions to suppress the installation of the wall program and its man page as a maintained version of this program is installed by Util-linux-ng.
  - [randy] - Added commands to the Chapter 6 Binutils instructions to suppress the installation of standards.info. Thanks to Greg Schafer for contributing the fix.
  - [randy] - Added a patch to the Procps instructions to fix a unicode related issue in the watch program.
  - [randy] - Added documentation installation commands to the Chapter 6 Kbd instructions.
  - [randy] - Modified the IPRoute2 installation command so that the docs are installed in a versioned directory.
  - [randy] - Modified the Groff installation command so that the docs are installed in a standardized versioned directory.
  - [randy] - Added documentation installation commands to the Chapter 6 Gawk instructions.
  - [randy] - Added commands to the Chapter 6 Flex instructions to install a .pdf doc file.
  - [randy] - Added a parameter to the configure command in the Automake instructions so that docs are installed in a versioned directory.
  - [randy] - Updated Module-Init-Tools to 3.4.1.
  - [randy] - Added documentation installation commands to the Chapter 6 Readline instructions.
  - [randy] - Added documentation installation commands to the Chapter 6 Ncurses instructions.
- 2008-10-10
  - [randy] - Added documentation enhancements to the E2fsprogs package.
  - [randy] - Removed an unnecessary parameter from the Util-linux-ng Chapter 6 make command. Thanks to Greg Schafer for pointing it out.
  - [randy] - Updated the Perl instructions. Thanks to Greg Schafer for pointing out the issues. This change also required that the Zlib package is built right before the Perl package in Chapter 6.
  - [randy] - Updated Vim to 7.2.
  - [randy] - Updated Udev to 130.
- 2008-10-09
  - [randy] - Updated File to 4.26
  - [randy] - Updated Shadow to 4.1.2.1.
  - [randy] - Updated Man-DB to 2.5.2.
  - [randy] - Updated Iproute to 2.6.26.
  - [randy] - Added a command to the Inetutils instructions to correct an issue with GCC-4.3.2.

- 2008-10-07
  - [randy] - Updated Autoconf to 2.63.
  - [randy] - Updated Libtool to 2.2.6a.
  - [randy] - Corrected the instruction to untar the E2fsprogs tarball in Section 2.3. Thanks to William Immendorf for pointing out the error.
  - [randy] - Updated Berkeley DB to 4.7.25.
  - [randy] - Updated Man-pages to 3.11.
  - [randy] - Updated Util-linux-ng to 2.14.1.
  - [randy] - Updated Texinfo to 4.13.
- 2008-10-06
  - [robert] - Added -v to the cp command in the Chapter 5 Expect instructions.
  - [randy] - Updated Tar to 1.20.
  - [randy] - Updated Perl to 5.10.0.
  - [randy] - Updated M4 to 1.4.11 and added it to the Chapter 5 build as it is required by the GMP package in Chapter 6.
  - [randy] - Updated Findutils to 4.4.0.
- 2008-10-05
  - [randy] - Updated E2fsprogs to 1.41.1.
  - [randy] - Added the Mktemp-1.5 package to the list of removed items in the Chapter3 'What's new ...' page.
  - [randy] - Updated Coreutils to 6.12. Thanks to William Immendorf for contributing a patch to add the mktemp program information to the Coreutils page.
  - [randy] - Updated the Bash Fixes patch to the -8 version.
  - [randy] - Added a patch to the Expect instructions to fix an issue with recent Tcl versions.
  - [randy] - Updated Tcl to 8.5.4.
  - [randy] - Updated the Linux kernel to 2.6.26.5.
  - [randy] - Updated Glibc to a 2.8 snapshot taken on 9/29/2008. The tarball of this snapshot includes the libidn data that previously was separately packaged.
  - [randy] - Added the GMP and MPFR packages to the list of packages in Chapter 3. Thanks to Lefteris Dimitroulakis for pointing out the omission.
- 2008-10-03
  - [bdubbs] - Added version check for Perl in Host System Requirements.
  - [randy] - Updated GCC to 4.3.2 which includes adding the GMP-4.2.4 and MPFR-2.3.2 packages. This new version of GCC requires the added packages. Thanks to DJ Lucas for the stimulus and initial work resulting in this and all of the other package updates coming up.
- 2008-07-11
  - [ken] - Belatedly fixed known vulnerabilities in perl.
- 2008-06-03
  - [bdubbs] - Added udev-config scripts to appendices.

  - [bdubbs] - Added lfs-bootscripts to appendices.
  - [bdubbs] - Updated license to Creative Commons with extracted code under the MIT license.
- 2008-05-23
  - [bryan] - Install a few extra rules from the etc/udev/packages directory in udev. Thanks to Dan Nicholson for noticing the issue.
- 2008-05-22
  - [bryan] - Updated Udev to 122, udev-config to 20080522, and lfs-bootscripts to 20080522. Also made persistent-net rules able to be pre-generated, using udevadm test. Fixes #2057, #2079 (I think), #2170, and #2186.
- 2008-04-23
  - [jhuntwork] - Use -mtune=native for glibc. We don't want our libc optimized for 486. It should be optimized for the local machine.
  - [jhuntwork] - Updated Autoconf to 2.62.
  - [jhuntwork] - Updated E2fsprogs to 1.40.8. Fixes #2173.
  - [jhuntwork] - Fixed behavior in kbd where man pages for optional programs that aren't built are installed. Thanks Greg Schafer for spotting this.
  - [jhuntwork] - Fixed kbd to install getkeycodes, setkeycodes and resizecons. Also moved loadkeys to /bin from /usr/bin. Thanks, Greg Schafer.
- 2008-04-22
  - [jhuntwork] - Updated Kbd to 1.14.1. Fixes #2162.
  - [jhuntwork] - Updated Flex to 2.5.35. Fixes #2179.
- 2008-04-11
  - [bdubbs] - Updated host requirments to check for symbolic links from sh, awk, and yacc.
- 2008-04-03
  - [jhuntwork] - Suppress installation of uptime in coreutils. Thanks to Randy McMurchy. Fixes #2133.
  - [jhuntwork] - Upgraded to iana-etc-2.30. Fixes #2174.
  - [jhuntwork] - Added patch for 256-byte inode support in GRUB. Fixes #2161.
- 2008-04-02
  - [jhuntwork] - Updated to linux-2.6.24.4, fixes #2157.
  - [jhuntwork] - Added an upstream patch for db-4.6.21, thanks Randy McMurchy for the report. Fixes #2164.
- 2008-03-30
  - [dnicholson] - Added `--sysconfdir` parameter to Man-db's configure command so that `man_db.conf` is installed in `/etc`.
- 2008-03-27
  - [ken] - Updated bzip2 to 1.0.5, fixes CVE-2008-1372.
- 2008-02-26
  - [ken] - Corrected typo in name of ru-ms keymap.

  - [ken] - Updated Kbd to 1.13.
- 2008-02-24
  - [matthew] - Add --libexecdir parameter to Man-db's configure command so that **globbing** and **manconv** are installed into /usr/libexec/man-db. Fixes *#2153*. Also, remove the --enable-mb-groff parameter, as this is now detected automatically.
- 2008-02-19
  - [ken] - Updated Grep to 2.5.3, thanks to Matthew for the fix for automated builds.
  - [ken] - Updated Flex to 2.5.34.
  - [ken] - Updated Module-Init-Tools to 3.4.
- 2008-02-17
  - [matthew] - Upgraded to latest upstream Vim patches.
  - [matthew] - Upgraded to Tcl-8.4.18. Fixes *#2146*.
  - [matthew] - Upgraded to Man-pages-2.78. Fixes *#2152*.
  - [matthew] - Upgraded to Man-DB-2.5.1. Fixes *#2148*.
  - [matthew] - Upgraded to Linux-2.6.24.2. Fixes *#2147*.
  - [matthew] - Now that **mktemp** is installed by Coreutils in chapter 5, there is no need to fix up GCC's **gccbug** in chapter 6. Thanks to Greg Schafer for the report.
  - [matthew] - Upgraded to Findutils-4.2.33. Fixes *#2151*.
  - [matthew] - Upgraded to E2fsprogs-1.40.6. Fixes *#2149*.
- 2008-02-07
  - [matthew] - Added a patch to fix a known issue in the Automake test suite. Fixes *#2143*.
  - [matthew] - Upgraded to Man-pages-2.77. Fixes *#2142*.
  - [matthew] - Upgraded to Libtool-1.5.26. Fixes *#2141*.
  - [matthew] - Upgraded to GCC-4.2.3. Fixes *#2140*.
  - [matthew] - Upgraded to Coreutils-6.10. Removed Mktemp-1.5 as Coreutils provides its own implementation now. Removed the coreutils binary suppression patch as the configure script can now be given a list of programs not to install. Fixes *#2133*.
  - [matthew] - Upgraded to E2fsprogs-1.40.5. Fixes *#2138*.
- 2008-01-29
  - [matthew] - Upgraded to Linux-2.6.24. Fixes *#2137*.
  - [matthew] - Upgraded to Findutils-4.2.32. Fixes *#2136*.
  - [matthew] - Upgraded to Automake-1.10.1. Fixes *#2132*.
- 2008-01-22
  - [matthew] - Replaced Util-Linux-2.12r, with Util-Linux-NG-2.13.1. Fixes *#2077*.
  - [matthew] - Upgraded to Tcl-8.4.17. Fixes *#2131*.
  - [matthew] - Upgraded to Man-Pages-2.76. Fixes *#2129*.
  - [matthew] - Upgraded to Linux-2.6.23.14. Fixes *#2128*.

- 2008-01-19
  - [matthew] - Add Perl to the list of host requirements, as it is required by Glibc. Thanks to Ben Collver for the report. Fixes *#2112*.
  - [matthew] - Mention **strace** as another means of logging installed files, and correct the URL of the Linux Standard Base specifications. Fixes *#2073* and *#2130*.
- 2008-01-04
  - [matthew] - Upgraded to latest upstream fixes for Vim.
  - [matthew] - Upgraded to Less-418. Fixes *#2124*.
  - [matthew] - Upgraded to File-4.23. Fixes *#2125*.
  - [matthew] - Upgraded to E2fsprogs-1.40.4. Fixes *#2123*.
- 2007-12-23
  - [matthew] - Upgraded to latest upstream fixes for Readline. Fixes *#2122*.
  - [matthew] - Upgraded to Man-Pages-2.74. Fixes *#2119*.
  - [matthew] - Upgraded to Linux-2.6.23.12. Fixes *#2118*.
  - [matthew] - Upgraded to latest upstream fixes for Bash. Fixes *#2121*.
- 2007-12-08
  - [matthew] - Upgraded to latest upstream fixes for Vim. Fixes *#2108*.
  - [matthew] - Upgraded to Texinfo-4.11. Fixes *#2074*.
  - [matthew] - Upgraded to Psmisc-22.6. Fixes *#2104*.
  - [matthew] - Upgraded to Man-Pages-2.70. Fixes *#2110*.
  - [matthew] - Upgraded to Man-DB-2.5.0. Fixes *#2109*.
  - [matthew] - Upgraded to Linux-2.6.23.9. Fixes *#2106*.
  - [matthew] - Upgraded to Less-416. Fixes *#2105*.
  - [matthew] - Upgraded to Gettext-0.17. Fixes *#2103*.
  - [matthew] - Removed the modifications to Gawk's config.h as Gawk-3.1.6 fixes the bug that they were working around. Fixes *#2107*. Thanks to Erik-Jan for the report.
  - [matthew] - Removed the modifications to Gawk's config.h as Gawk-3.1.6 fixes the bug that they were working around. Fixes *#2107*. Thanks to Erik-Jan for the report.
  - [matthew] - Upgraded to E2fsprogs-1.40.3. Fixes *#2116*.
- 2007-11-25
  - [bdubbs] - Fixed test for Debian binutils.
- 2007-10-29
  - [bdubbs] - Removed obsolete note from Creating Symlinks section about continuation lines in udev rules. Changed dailout group to uucp for udev rule compatability.
  - [matthew] - Upgrade to the latest upstream patches for Vim.
  - [matthew] - Add a patch to fix a segfault in usb_id.
  - [matthew] - Upgrade to Tcl-8.4.16. Fixes *#2084*.

  - [matthew] - Upgrade to Tar-1.19. Fixes *#2090*.
  - [matthew] - Upgrade to Man-Pages-2.67. Fixes *#2078*.
  - [matthew] - Upgrade to Linux-2.6.23.1. Fixes *#2088*.
  - [matthew] - Upgrade to Less-409. Fixes *#2087*.
  - [matthew] - Upgrade to IPRoute2-2.6.23. Fixes *#2091*.
  - [matthew] - Upgrade to Glibc-2.7. Fixes *#2095*.
  - [matthew] - Upgrade to GCC-4.2.2. Fixes *#2089*.
  - [matthew] - Upgrade to Gawk-3.1.6. Fixes *#2098*.
  - [matthew] - Upgrade to DB-4.6.21. Fixes *#2086*.
- 2007-09-25
  - [manuel] - More updates in dependencies list. Thanks to Chris Staub for the patch.
- 2007-09-23
  - [manuel] - Updated dependencies list. Thanks to Chris Staub for the patch.
- 2007-09-21
  - [manuel] - Fixed glibc-libidn tarball extension.
- 2007-09-18
  - [manuel] - Added remap attributes to userinput tags in packages pages to help adding package manager support and other extensions into jhalfs. Made all testsuite commands screen blocks for consistency.
- 2007-09-16
  - [manuel] - Updated Ncurses contents list and fixes some typos. Thanks to Chris Staub for the patch.
- 2007-09-15
  - [matthew] - Add latest upstream patches for Vim.
  - [matthew] - Upgrade to Sysklogd-1.5. Fixes *#2055*.
  - [matthew] - Add latest upstream patches for Readline. Fixes *#2068*.
  - [matthew] - Upgrade to Man-pages 2.64. Fixes *#2061*.
  - [matthew] - Upgrade to Linux-2.6.22.6. Fixes *#2070*.
  - [jhuntwork] - Upgrade to Glibc-2.6.1. Fixes *#2018*. Thanks to Matthew Burgess for preparing a discrete patch, Robert Connolly and Dan Nicholson for investigating how best to adjust CFLAGS, and Greg Schafer for showing the technical benefits of using CFLAGS with Glibc.
  - [jhuntwork] - Upgrade to GCC-4.2.1. Fixes *#2002*. Thanks to Matthew Burgess for preparing a discrete patch.
  - [matthew] - Upgrade to DB-4.6.19. Fixes *#2051*.
  - [matthew] - Upgrade to Binutils-2.18. Fixes *#2069*.
  - [matthew] - Add latest upstream patches for Bash. Fixes *#2067*.
- 2007-09-07
  - [manuel] - Added sect1info metainformation blocks to packages pages to help adding package manager support into jhalfs.

LFS 6.3 released August 28, 2007.

1.4. Resources

1.4.1. FAQ

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

1.4.2. Mailing Lists

The `linuxfromscratch.org` server hosts a number of mailing lists used for the development of the LFS project. These lists include the main development and support lists, among others. If the FAQ does not solve the problem you are having, the next step would be to search the mailing lists at *http://www.linuxfromscratch.org/search.html*.

For information on the different lists, how to subscribe, archive locations, and additional information, visit *http://www.linuxfromscratch.org/mail.html*.

1.4.3. IRC

Several members of the LFS community offer assistance on our community Internet Relay Chat (IRC) network. Before using this support, please make sure that your question is not already answered in the LFS FAQ or the mailing list archives. You can find the IRC network at `irc.linuxfromscratch.org`. The support channel is named #LFS-support.

1.4.4. Mirror Sites

The LFS project has a number of world-wide mirrors to make accessing the website and downloading the required packages more convenient. Please visit the LFS website at *http://www.linuxfromscratch.org/mirrors.html* for a list of current mirrors.

1.4.5. Contact Information

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

1.5. Help

If an issue or a question is encountered while working through this book, check the FAQ page at *http://www.linuxfromscratch.org/faq/#generalfaq*. Questions are often already answered there. If your question is not answered on this page, try to find the source of the problem. The following hint will give you some guidance for troubleshooting: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

If you cannot find your problem listed in the FAQ, search the mailing lists at *http://www.linuxfromscratch.org/search.html*.

We also have a wonderful LFS community that is willing to offer assistance through the mailing lists and IRC (see the Section 1.4, “Resources” section of this book). However, we get several support questions every day and many of them can be easily answered by going to the FAQ and by searching the mailing lists first. So, for us to offer the best assistance possible, you need to do some research on your own first. That allows us to focus on the more unusual support needs. If your searches do not produce a solution, please include all relevant information (mentioned below) in your request for help.

1.5.1. Things to Mention

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

- The version of the book being used (in this case 6.4)
- The host distribution and version being used to create LFS
- The package or section the problem was encountered in
- The exact error message or symptom being received
- Note whether you have deviated from the book at all

> **Note**
>
> Deviating from this book does *not* mean that we will not help you. After all, LFS is about personal preference. Being upfront about any changes to the established procedure helps us evaluate and determine possible causes of your problem.

1.5.2. Configure Script Problems

If something goes wrong while running the **configure** script, review the `config.log` file. This file may contain errors encountered during **configure** which were not printed to the screen. Include the *relevant* lines if you need to ask for help.

1.5.3. Compilation Problems

Both the screen output and the contents of various files are useful in determining the cause of compilation problems. The screen output from the **configure** script and the **make** run can be helpful. It is not necessary to include the entire output, but do include enough of the relevant information. Below is an example of the type of information to include from the screen output from **make**:

```
gcc -DALIASPATH=\"/mnt/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 would just include the bottom section:

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

This is not enough information to properly diagnose the problem because it only notes that something went wrong, not *what* went wrong. The entire section, as in the example above, is what should be saved because it includes the command that was executed and the associated error message(s).

An excellent article about asking for help on the Internet is available online at *http://catb.org/~esr/faqs/smart-questions.html*. Read and follow the hints in this document to increase the likelihood of getting the help you need.

Part II. Preparing for the Build

Chapter 2. Preparing a New Partition

2.1. Introduction

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

2.2. Creating a New Partition

Like most other operating systems, LFS is usually installed on a dedicated partition. The recommended approach to building an LFS system is to use an available empty partition or, if you have enough unpartitioned space, to create one. However, an LFS system (in fact even multiple LFS systems) may also be installed on a partition already occupied by another operating system and the different systems will co-exist peacefully. The document *http://www.linuxfromscratch.org/hints/downloads/files/lfs_next_to_existing_systems.txt* explains how to implement this, whereas this book discusses the method of using a fresh partition for the installation.

A minimal system requires a partition of around 1.3 gigabytes (GB). This is enough to store all the source tarballs and compile the packages. However, if the LFS system is intended to be the primary Linux system, additional software will probably be installed which will require additional space (2-3 GB). The LFS system itself will not take up this much room. A large portion of this requirement is to provide sufficient free temporary storage. Compiling packages can require a lot of disk space which will be reclaimed after the package is installed.

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

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

Remember the designation of the new partition (e.g., `hda5`). This book will refer to this as the LFS partition. Also remember the designation of the `swap` partition. These names will be needed later for the `/etc/fstab` file.

2.3. Creating a File System on the Partition

Now that a blank partition has been set up, the file system can be created. The most widely-used system in the Linux world is the second extended file system (`ext2`), but with newer high-capacity hard disks, journaling file systems are becoming increasingly popular. The third extended filesystem (`ext3`) is a widely used enhancement to `ext2`, which adds journalling capabilities and is compatible with the E2fsprogs utilities. We will create an `ext3` file system. Instructions for creating other file systems can be found at *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html*.

To create an `ext3` file system on the LFS partition, run the following:

```
mke2fs -jv /dev/<xxx>
```

Replace `<xxx>` with the name of the LFS partition (`hda5` in our previous example).

Note

Some host distributions use custom features in their filesystem creation tools (E2fsprogs). This can cause problems when booting into your new LFS in Chapter 9, as those features will not be supported by the LFS-installed E2fsprogs; you will get an error similar to “unsupported filesystem features, upgrade your e2fsprogs”. To check if your host system uses custom enhancements, run the following command:

```
debugfs -R feature /dev/<xxx>
```

If the output contains features other than `has_journal`, `ext_attr`, `resize_inode`, `dir_index`, `filetype`, `sparse_super`, `large_file` or `needs_recovery`, then your host system may have custom enhancements. In that case, to avoid later problems, you should compile the stock E2fsprogs package and use the resulting binaries to re-create the filesystem on your LFS partition:

```
cd /tmp
tar -xzvf /path/to/sources/e2fsprogs-1.41.3.tar.gz
cd e2fsprogs-1.41.3
mkdir -v build
cd build
../configure
make #note that we intentionally don't 'make install' here!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.41.3
```

If you are using an existing `swap` partition, there is no need to format it. If a new `swap` partition was created, it will need to be initialized with this command:

```
mkswap /dev/<yyy>
```

Replace <*yyy*> with the name of the `swap` partition.

2.4. Mounting the New Partition

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

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

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

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

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
```

Replace <*xxx*> with the designation of the LFS partition.

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

```
mkdir -pv $LFS
mount -v -t ext3 /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext3 /dev/<yyy> $LFS/usr
```

Replace `<xxx>` and `<yyy>` with the appropriate partition names.

Ensure that this new partition is not mounted with permissions that are too restrictive (such as the `nosuid`, `nodev`, or `noatime` options). Run the **mount** command without any parameters to see what options are set for the mounted LFS partition. If `nosuid`, `nodev`, and/or `noatime` are set, the partition will need to be remounted.

If you are using a `swap` partition, ensure that it is enabled using the **swapon** command:

```
/sbin/swapon -v /dev/<zzz>
```

Replace `<zzz>` with the name of the `swap` partition.

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

Chapter 3. Packages and Patches

3.1. Introduction

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

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

Downloaded packages and patches will need to be stored somewhere that is conveniently available throughout the entire build. A working directory is also required to unpack the sources and build them. `$LFS/sources` can be used both as the place to store the tarballs and patches and as a working directory. By using this directory, the required elements will be located on the LFS partition and will be available during all stages of the building process.

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

```
mkdir -v $LFS/sources
```

Make this directory writable and sticky. “Sticky” means that even if multiple users have write permission on a directory, only the owner of a file can delete the file within a sticky directory. The following command will enable the write and sticky modes:

```
chmod -v a+wt $LFS/sources
```

3.2. All Packages

Download or otherwise obtain the following packages:

- **Autoconf (2.63) - 1,195 KB:**

  Home page: *http://www.gnu.org/software/autoconf/*

  Download: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.63.tar.bz2*

  MD5 sum: `7565809ed801bb5726da0631ceab3699`

- **Automake (1.10.1) - 897 KB:**

  Home page: *http://www.gnu.org/software/automake/*

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

  MD5 sum: `4510391e6b3edaa4cffb3ced87c9560c`

- **Bash (3.2) - 2,471 KB:**

  Home page: *http://www.gnu.org/software/bash/*

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

  MD5 sum: `00bfa16d58e034e3c2aa27f390390d30`

- **Bash Documentation (3.2) - 2,143 KB:**

Download: *http://ftp.gnu.org/gnu/bash/bash-doc-3.2.tar.gz*
MD5 sum: `0e904cb46ca873fcfa65df19b024bec9`

- **Berkeley DB (4.7.25) - 13,124 KB:**

Home page: *http://www.oracle.com/technology/software/products/berkeley-db/index.html*
Download: *http://download-east.oracle.com/berkeley-db/db-4.7.25.tar.gz*
MD5 sum: `ec2b87e833779681a0c3a814aa71359e`

- **Binutils (2.18) - 14,612 KB:**

Home page: *http://sources.redhat.com/binutils/*
Download: *http://ftp.gnu.org/gnu/binutils/binutils-2.18.tar.bz2*
MD5 sum: `9d22ee4dafa3a194457caf4706f9cf01`

- **Bison (2.3) - 1,055 KB:**

Home page: *http://www.gnu.org/software/bison/*
Download: *http://ftp.gnu.org/gnu/bison/bison-2.3.tar.bz2*
MD5 sum: `c18640c6ec31a169d351e3117ecce3ec`

- **Bzip2 (1.0.5) - 8,228 KB:**

Home page: *http://www.bzip.org/*
Download: *http://www.bzip.org/1.0.5/bzip2-1.0.5.tar.gz*
MD5 sum: `3c15a0c8d1d3ee1c46a1634d00617b1a`

- **Coreutils (6.12) - 9,001 KB:**

Home page: *http://www.gnu.org/software/coreutils/*
Download: *http://ftp.gnu.org/gnu/coreutils/coreutils-6.12.tar.gz*
MD5 sum: `2ca9ac69823dbd567b905a9e9f53c4f6`

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

Home page: *http://www.gnu.org/software/dejagnu/*
Download: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.4.4.tar.gz*
MD5 sum: `053f18fd5d00873de365413cab17a666`

- **Diffutils (2.8.1) - 762 KB:**

Home page: *http://www.gnu.org/software/diffutils/*
Download: *http://ftp.gnu.org/gnu/diffutils/diffutils-2.8.1.tar.gz*
MD5 sum: `71f9c5ae19b60608f6c7f162da86a428`

- **E2fsprogs (1.41.3) - 4,276 KB:**

Home page: *http://e2fsprogs.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.41.3.tar.gz*
MD5 sum: `b21d26fc46c584021dc9c444933ee1c2`

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

Home page: *http://expect.nist.gov/*
Download: *http://expect.nist.gov/src/expect-5.43.0.tar.gz*
MD5 sum: `43e1dc0e0bc9492cf2e1a6f59f276bc3`

• **File (4.26) - 584 KB:**
Home page: *http://www.darwinsys.com/file/*
Download: *ftp://ftp.astron.com/pub/file/file-4.26.tar.gz*
MD5 sum: `74cd5466416136da30a4e69f74dbc7a0`

> **Note**
> File (4.26) may no longer be available at the listed location. The site administrators of the master download location occasionally remove older versions when new ones are released. An alternative download location that may have the correct version available can also be found at: *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• **Findutils (4.4.0) - 2,029 KB:**
Home page: *http://www.gnu.org/software/findutils/*
Download: *http://ftp.gnu.org/gnu/findutils/findutils-4.4.0.tar.gz*
MD5 sum: `49e769ac4382fae6f104f99d54d0a112`

• **Flex (2.5.35) - 1,229 KB:**
Home page: *http://flex.sourceforge.net*
Download: *http://prdownloads.sourceforge.net/flex/flex-2.5.35.tar.bz2*
MD5 sum: `10714e50cea54dc7a227e3eddcd44d57`

• **Gawk (3.1.6) - 1,818 KB:**
Home page: *http://www.gnu.org/software/gawk/*
Download: *http://ftp.gnu.org/gnu/gawk/gawk-3.1.6.tar.bz2*
MD5 sum: `c9926c0bc8c177cb9579708ce67f0d75`

• **GCC (4.3.2) - 58,929 KB:**
Home page: *http://gcc.gnu.org/*
Download: *http://ftp.gnu.org/gnu/gcc/gcc-4.3.2/gcc-4.3.2.tar.bz2*
MD5 sum: `5dfac5da961ecd5f227c3175859a486d`

• **Gettext (0.17) - 11,368 KB:**
Home page: *http://www.gnu.org/software/gettext/*
Download: *http://ftp.gnu.org/gnu/gettext/gettext-0.17.tar.gz*
MD5 sum: `58a2bc6d39c0ba57823034d55d65d606`

• **Glibc (2.8-20080929) - 16,231 KB:**
Home page: *http://www.gnu.org/software/libc/*
Download: *ftp://sources.redhat.com/pub/glibc/snapshots/glibc-2.8-20080929.tar.bz2*
MD5 sum: `ef223822e84f38dc6b3762bcf3bd6c5e`

• **GMP (4.2.4) - 1,170 KB:**
Home page: *http://www.gnu.org/software/gmp/*
Download: *http://ftp.gnu.org/gnu/gmp/gmp-4.2.4.tar.bz2*
MD5 sum: `fc1e3b3a2a5038d4d74138d0b9cf8dbe`

• **Grep (2.5.3) - 604 KB:**
Home page: *http://www.gnu.org/software/grep/*
Download: *http://ftp.gnu.org/gnu/grep/grep-2.5.3.tar.bz2*
MD5 sum: `27061ce1fde82876970b6549a156da8b`

- **Groff (1.18.1.4) - 2,265 KB:**
Home page: *http://www.gnu.org/software/groff/*
Download: *http://ftp.gnu.org/gnu/groff/groff-1.18.1.4.tar.gz*
MD5 sum: `ceecb81533936d251ed015f40e5f7287`
- **GRUB (0.97) - 950 KB:**
Home page: *http://www.gnu.org/software/grub/*
Download: *ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz*
MD5 sum: `cd3f3eb54446be6003156158d51f4884`
- **Gzip (1.3.12) - 451 KB:**
Home page: *http://www.gzip.org/*
Download: *http://ftp.gnu.org/gnu/gzip/gzip-1.3.12.tar.gz*
MD5 sum: `b5bac2d21840ae077e0217bc5e4845b1`
- **Iana-Etc (2.30) - 204 KB:**
Home page: *http://sethwklein.net/iana-etc*
Download: *http://sethwklein.net/iana-etc-2.30.tar.bz2*
MD5 sum: `3ba3afb1d1b261383d247f46cb135ee8`
- **Inetutils (1.5) - 1,357 KB:**
Home page: *http://www.gnu.org/software/inetutils/*
Download: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.5.tar.gz*
MD5 sum: `aeacd11d19bf25c89d4eff38346bdfb9`
- **IPRoute2 (2.6.26) - 359 KB:**
Home page: *http://linux-net.osdl.org/index.php/Iproute2*
Download: *http://developer.osdl.org/dev/iproute2/download/iproute2-2.6.26.tar.bz2*
MD5 sum: `7d221e735cba05709341cd46401c4ecd`
- **Kbd (1.14.1) - 989 KB:**
Download: *http://ftp.altlinux.com/pub/people/legion/kbd/kbd-1.14.1.tar.gz*
MD5 sum: `0f4e474032c992c05650924f29a06a92`
- **Less (418) - 292 KB:**
Home page: *http://www.greenwoodsoftware.com/less/*
Download: *http://www.greenwoodsoftware.com/less/less-418.tar.gz*
MD5 sum: `b5864d76c54ddf4627fd57ab333c88b4`
- **LFS-Bootscripts (20081031) - 42 KB:**
Download: *http://www.linuxfromscratch.org/lfs/downloads/6.4/lfs-bootscripts-20081031.tar.bz2*
MD5 sum: `9cbb57200f7e3db179afb1f023379e22`
- **Libtool (2.2.6a) - 2,870 KB:**
Home page: *http://www.gnu.org/software/libtool/*
Download: *http://ftp.gnu.org/gnu/libtool/libtool-2.2.6a.tar.gz*
MD5 sum: `8ca1ea241cd27ff9832e045fe9afe4fd`

- **Linux (2.6.27.4) - 49,232 KB:**
Home page: *http://www.kernel.org/*
Download: *http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.27.4.tar.bz2*
MD5 sum: `3880fe9f19b9a7690afd151326eb7ce5`

> **Note**
> The Linux kernel is updated relatively often, many times due to discoveries of security vulnerabilities. The latest available 2.6.27.x kernel version should be used, unless the errata page says otherwise.
> For users with limited speed or expensive bandwidth who wish to update the Linux kernel, a baseline version of the package and patches can be downloaded separately. This may save some time or cost for a subsequent patch level upgrade within a minor release.

- **M4 (1.4.12) - 884 KB:**
Home page: *http://www.gnu.org/software/m4/*
Download: *http://ftp.gnu.org/gnu/m4/m4-1.4.12.tar.bz2*
MD5 sum: `b3587c993523dd320c318ec456876839`
- **Make (3.81) - 1,125 KB:**
Home page: *http://www.gnu.org/software/make/*
Download: *http://ftp.gnu.org/gnu/make/make-3.81.tar.bz2*
MD5 sum: `354853e0b2da90c527e35aabb8d6f1e6`
- **Man-DB (2.5.2) - 1,772 KB:**
Home page: *http://www.nongnu.org/man-db/*
Download: *http://download.savannah.gnu.org/releases/man-db/man-db-2.5.2.tar.gz*
MD5 sum: `9529aadae273566a170dee4e18aad6c1`
- **Man-pages (3.11) - 987 KB:**
Download: *http://www.kernel.org/pub/linux/docs/manpages/Archive/man-pages-3.11.tar.bz2*
MD5 sum: `f66e01df3a22e18d25c5865925dd9288`
- **Module-Init-Tools (3.4.1) - 195 KB:**
Home page: *http://www.kerneltools.org/KernelTools.org*
Download: *http://www.kernel.org/pub/linux/utils/kernel/module-init-tools/module-init-tools-3.4.1.tar.bz2*
MD5 sum: `e253b066a1bab1d727ca0d54f001b49c`
- **MPFR (2.3.2) - 986 KB:**
Home page: *http://www.mpfr.org/*
Download: *http://www.mpfr.org/mpfr-current/mpfr-2.3.2.tar.bz2*
MD5 sum: `527147c097874340cb9cee0579dacf3b`
- **Ncurses (5.6) - 2,346 KB:**
Home page: *http://www.gnu.org/software/ncurses/*
Download: *ftp://ftp.gnu.org/gnu/ncurses/ncurses-5.6.tar.gz*
MD5 sum: `b6593abe1089d6aab1551c105c9300e3`
- **Patch (2.5.4) - 183 KB:**
Home page: *http://www.gnu.org/software/patch/*
Download: *http://ftp.gnu.org/gnu/patch/patch-2.5.4.tar.gz*
MD5 sum: `ee5ae84d115f051d87fcaaef3b4ae782`

- **Perl (5.10.0) - 15,595 KB:**
Home page: *http://cpan.org/*
Download: *http://cpan.org/src/perl-5.10.0.tar.gz*
MD5 sum: `d2c39b002ebfd2c3c5dba589365c5a71`
- **Procps (3.2.7) - 275 KB:**
Home page: *http://procps.sourceforge.net/*
Download: *http://procps.sourceforge.net/procps-3.2.7.tar.gz*
MD5 sum: `f490bca772b16472962c7b9f23b1e97d`
- **Psmisc (22.6) - 277 KB:**
Home page: *http://psmisc.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/psmisc/psmisc-22.6.tar.gz*
MD5 sum: `2e81938855cf5cc38856bd4a31d79a4c`
- **Readline (5.2) - 1,990 KB:**
Home page: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
Download: *http://ftp.gnu.org/gnu/readline/readline-5.2.tar.gz*
MD5 sum: `e39331f32ad14009b9ff49cc10c5e751`
- **Sed (4.1.5) - 781 KB:**
Home page: *http://www.gnu.org/software/sed/*
Download: *http://ftp.gnu.org/gnu/sed/sed-4.1.5.tar.gz*
MD5 sum: `7a1cbbbb3341287308e140bd4834c3ba`
- **Shadow (4.1.2.1) - 1,697 KB:**
Home page: *http://pkg-shadow.alioth.debian.org/*
Download: *ftp://pkg-shadow.alioth.debian.org/pub/pkg-shadow/shadow-4.1.2.1.tar.bz2*
MD5 sum: `c178e49c45495e296dabbe4ae01a0fbe`
- **Sysklogd (1.5) - 85 KB:**
Home page: *http://www.infodrom.org/projects/sysklogd/*
Download: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz*
MD5 sum: `e053094e8103165f98ddafe828f6ae4b`
- **Sysvinit (2.86) - 97 KB:**
Download: *ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/sysvinit-2.86.tar.gz*
MD5 sum: `7d5d61c026122ab791ac04c8a84db967`
- **Tar (1.20) - 1,912 KB:**
Home page: *http://www.gnu.org/software/tar/*
Download: *http://ftp.gnu.org/gnu/tar/tar-1.20.tar.bz2*
MD5 sum: `1a7e17f27abf583b3b0bc059a827e68b`
- **Tcl (8.5.5) - 4,316 KB:**
Home page: *http://tcl.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/tcl/tcl8.5.5-src.tar.gz*
MD5 sum: `39faed045bd03da1267fb66c9b75349f`
- **Texinfo (4.13a) - 2,751 KB:**
Home page: *http://www.gnu.org/software/texinfo/*
Download: *http://ftp.gnu.org/gnu/texinfo/texinfo-4.13a.tar.gz*
MD5 sum: `71ba711519209b5fb583fed2b3d86fcb`

- **Udev (130) - 442 KB:**
Home page: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html*
Download: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-130.tar.bz2*
MD5 sum: `eaaac3c45b8c87d81a82fed254ecee25`

- **Udev Configuration Tarball - 13 KB:**
Download: *http://www.linuxfromscratch.org/lfs/downloads/6.4/udev-config-20081015.tar.bz2*
MD5 sum: `ac53f85ee6d3f31964a591f4b86e7707`

- **Util-linux-ng (2.14.1) - 2,929 KB:**
Home page: *http://userweb.kernel.org/~kzak/util-linux-ng/*
Download: *http://www.kernel.org/pub/linux/utils/util-linux-ng/v2.14/util-linux-ng-2.14.1.tar.bz2*
MD5 sum: `9aab772ee9b1f4e67dff98169f3cb380`

- **Vim (7.2) - 7,203 KB:**
Home page: *http://www.vim.org*
Download: *ftp://ftp.vim.org/pub/vim/unix/vim-7.2.tar.bz2*
MD5 sum: `f0901284b338e448bfd79ccca0041254`

- **Vim (7.2) language files (optional) - 1,365 KB:**
Home page: *http://www.vim.org*
Download: *ftp://ftp.vim.org/pub/vim/extra/vim-7.2-lang.tar.gz*
MD5 sum: `d8884786979e0e520c112faf2e176f05`

- **Zlib (1.2.3) - 416 KB:**
Home page: *http://www.zlib.net/*
Download: *http://www.zlib.net/zlib-1.2.3.tar.bz2*
MD5 sum: `dee233bf288ee795ac96a98cc2e369b6`

Total size of these packages: about 257 MB

3.3. Needed Patches

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

- **Automake Test Suite Patch - 3 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/automake-1.10.1-test_fix-1.patch*
MD5 sum: `4d8aa269951bb3cd876d2bb663cb04cc`

- **Bash Upstream Fixes Patch - 66 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/bash-3.2-fixes-8.patch*
MD5 sum: `7729e8bb1adb57c8d3c4c3a34a5bbab0`

- **Berkeley DB Upstream Fixes Patch - 1.9 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/db-4.7.25-upstream_fixes-1.patch*
MD5 sum: `dfe0d2a27439454fbafdeeef65fefade`

- **Binutils GCC 4.3 Patch - 1.1 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/binutils-2.18-GCC43-1.patch*
MD5 sum: `d77fa789b4cae8b1ef7bc10e6220a529`

- **Binutils Texinfo Version Patch - 1 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/binutils-2.18-configure-1.patch*
MD5 sum: `83877c299e3e3080952214e479396f23`
- **Bzip2 Documentation Patch - 1.6 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/bzip2-1.0.5-install_docs-1.patch*
MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`
- **Coreutils Internationalization Fixes Patch - 104 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/coreutils-6.12-i18n-2.patch*
MD5 sum: `2b6182f77f8b575e27d7743dd403104e`
- **Coreutils Old Kernel Patch - 3.3 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/coreutils-6.12-old_build_kernel-1.patch*
MD5 sum: `5e8622abe6c6d81901b910383c6fb611`
- **Coreutils Uname Patch - 4.6 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/coreutils-6.12-uname-1.patch*
MD5 sum: `c05b735710fbd62239588c07084852a0`
- **Diffutils Internationalization Fixes Patch - 18 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/diffutils-2.8.1-i18n-1.patch*
MD5 sum: `c8d481223db274a33b121fb8c25af9f7`
- **Expect Spawn Patch - 6.8 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/expect-5.43.0-spawn-1.patch*
MD5 sum: `ef6d0d0221c571fb420afb7033b3bbba`
- **Expect Tcl Patch - 4.1 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/expect-5.43.0-tcl_8.5.5_fix-1.patch*
MD5 sum: `6904a384960ce0e8f0d0b32f7903d7a1`
- **Glibc Iconv Test Fixes Patch - 1.7 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/glibc-2.8-20080929-iconv_tests-1.patch*
MD5 sum: `cc5e95e418e0b2f8a54b14cf90c7c3b2`
- **Glibc Ildoubl Test Fix Patch - 1.0 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/glibc-2.8-20080929-ildoubl_test-1.patch*
MD5 sum: `4dc864a487eee8426413542591d19edb`
- **Grep Debian Patch - 27 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/grep-2.5.3-debian_fixes-1.patch*
MD5 sum: `337d017202d7e3b08d428a89da3ee572`
- **Grep Upstream Fixes Patch - 5.8 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/grep-2.5.3-upstream_fixes-1.patch*
MD5 sum: `44f9c5e7df7746e6115be47e5a068ab8`
- **Groff Debian Patch - 379 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/groff-1.18.1.4-debian_fixes-1.patch*
MD5 sum: `05607e7fcfd6e5091f020bf44ddca10b`
- **GRUB Disk Geometry Patch - 28 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/grub-0.97-disk_geometry-1.patch*
MD5 sum: `bf1594e82940e25d089feca74c6f1879`

- **GRUB 256-Byte Inodes Patch - 4.8 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/grub-0.97-256byte_inode-1.patch*
MD5 sum: `2482bef9c1866b4045767a56268ba673`

- **Inetutils No-Server-Man-Pages Patch - 5.3 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/inetutils-1.5-no_server_man_pages-2.patch*
MD5 sum: `ec83aa00fb111f6f9d9aca04de9cb753`

- **Kbd Backspace/Delete Fix Patch - 13 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/kbd-1.14.1-backspace-1.patch*
MD5 sum: `fe51ec685687ce9d29463d786ba0c2d4`

- **Module-init-tools Man-Pages Patch - 35 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/module-init-tools-3.4.1-manpages-1.patch*
MD5 sum: `2271047586981ae23adf01cc13d97791`

- **Ncurses Coverity Patch - 16.8 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/ncurses-5.6-coverity_fixes-1.patch*
MD5 sum: `aa2fa9d0e89bbfdb4ce7e0e6b4b46670`

- **Perl Consolidated Patch - 7.1 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/perl-5.10.0-consolidated-1.patch*
MD5 sum: `d1bcffb5d671bd659f7ca5c451a0c752`

- **Procps Watch Patch - 3.6 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/procps-3.2.7-watch_unicode-1.patch*
MD5 sum: `2e5b57608177bd54349c718db9b5843d`

- **Readline Fixes Patch - 18 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/readline-5.2-fixes-5.patch*
MD5 sum: `7390b2296b7b11209829646537294ebb`

- **Vim Fixes Patch - 29.3 KB:**
Download: *http://www.linuxfromscratch.org/patches/lfs/6.4/vim-7.2-fixes-3.patch*
MD5 sum: `4b526f493995d2eb6fd415eb62ff43d8`

Total size of these patches: about 790.8 KB

In addition to the above required patches, there exist a number of optional patches created by the LFS community. These optional patches solve minor problems or enable functionality that is not enabled by default. Feel free to peruse the patches database located at *http://www.linuxfromscratch.org/patches/* and acquire any additional patches to suit the system needs.

Chapter 4. Final Preparations

4.1. About $LFS

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

```
echo $LFS
```

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

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

Having this variable set is beneficial in that commands such as **mkdir $LFS/tools** can be typed literally. The shell will automatically replace “$LFS” with “/mnt/lfs” (or whatever the variable was set to) when it processes the command line.

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

4.2. Creating the $LFS/tools Directory

All programs compiled in Chapter 5 will be installed under `$LFS/tools` to keep them separate from the programs compiled in Chapter 6. The programs compiled here are temporary tools and will not be a part of the final LFS system. By keeping these programs in a separate directory, they can easily be discarded later after their use. This also prevents these programs from ending up in the host production directories (easy to do by accident in Chapter 5).

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

```
mkdir -v $LFS/tools
```

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

```
ln -sv $LFS/tools /
```

Note

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

The created symlink enables the toolchain to be compiled so that it always refers to `/tools`, meaning that the compiler, assembler, and linker will work both in this chapter (when we are still using some tools from the host) and in the next (when we are “chrooted” to the LFS partition).

4.3. Adding the LFS User

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

```
groupadd lfs
useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```

The meaning of the command line options:

-s /bin/bash
: This makes **bash** the default shell for user `lfs`.

-g lfs
: This option adds user `lfs` to group `lfs`.

-m
: This creates a home directory for `lfs`.

-k /dev/null
: This parameter prevents possible copying of files from a skeleton directory (default is `/etc/skel`) by changing the input location to the special null device.

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

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

```
passwd lfs
```

Grant `lfs` full access to `$LFS/tools` by making `lfs` the directory owner:

```
chown -v lfs $LFS/tools
```

If a separate working directory was created as suggested, give user `lfs` ownership of this directory:

```
chown -v lfs $LFS/sources
```

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

```
su - lfs
```

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

4.4. Setting Up the Environment

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

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

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

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

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL PATH
EOF
```

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

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

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

The `LC_ALL` variable controls the localization of certain programs, making their messages follow the conventions of a specified country. If the host system uses a version of Glibc older than 2.2.4, having `LC_ALL` set to something other than “POSIX” or “C” (during this chapter) may cause issues if you exit the chroot environment and wish to return later. Setting `LC_ALL` to “POSIX” or “C” (the two are equivalent) ensures that everything will work as expected in the chroot environment.

By putting `/tools/bin` ahead of the standard `PATH`, all the programs installed in Chapter 5 are picked up by the shell immediately after their installation. This, combined with turning off hashing, limits the risk that old programs are used from the host when the same programs are available in the chapter 5 environment.

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

```
source ~/.bash_profile
```

4.5. About SBUs

Many people would like to know beforehand approximately how long it takes to compile and install each package. Because Linux From Scratch can be built on many different systems, it is impossible to provide accurate time estimates. The biggest package (Glibc) will take approximately 20 minutes on the fastest systems, but could take up to three days on slower systems! Instead of providing actual times, the Standard Build Unit (SBU) measure will be used instead.

The SBU measure works as follows. The first package to be compiled from this book is Binutils in Chapter 5. The time it takes to compile this package is what will be referred to as the Standard Build Unit or SBU. All other compile times will be expressed relative to this time.

For example, consider a package whose compilation time is 4.5 SBUs. This means that if a system took 10 minutes to compile and install the first pass of Binutils, it will take *approximately* 45 minutes to build this example package. Fortunately, most build times are shorter than the one for Binutils.

In general, SBUs are not entirely accurate because they depend on many factors, including the host system's version of GCC. Note that on Symmetric Multi-Processor (SMP)-based machines, SBUs are even less accurate. They are provided here to give an estimate of how long it might take to install a package, but the numbers can vary by as much as dozens of minutes in some cases.

To view actual timings for a number of specific machines, we recommend The LinuxFromScratch SBU Home Page at *http://www.linuxfromscratch.org/~sbu/*.

4.6. About the Test Suites

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

Some test suites are more important than others. For example, the test suites for the core toolchain packages—GCC, Binutils, and Glibc—are of the utmost importance due to their central role in a properly functioning system. The test suites for GCC and Glibc can take a very long time to complete, especially on slower hardware, but are strongly recommended.

Note

Experience has shown that there is little to be gained from running the test suites in Chapter 5. There can be no escaping the fact that the host system always exerts some influence on the tests in that chapter, often causing inexplicable failures. Because the tools built in Chapter 5 are temporary and eventually discarded, we do not recommend running the test suites in Chapter 5 for the average reader. The instructions for running those test suites are provided for the benefit of testers and developers, but they are strictly optional.

A common issue with running the test suites for Binutils and GCC is running out of pseudo terminals (PTYs). This can result in a high number of failing tests. This may happen for several reasons, but the most likely cause is that the host system does not have the `devpts` file system set up correctly. This issue is discussed in greater detail in Chapter 5.

Sometimes package test suites will fail, but for reasons which the developers are aware of and have deemed non-critical. Consult the logs located at *http://www.linuxfromscratch.org/lfs/build-logs/6.4/* to verify whether or not these failures are expected. This site is valid for all tests throughout this book.

Chapter 5. Constructing a Temporary System

5.1. Introduction

This chapter shows how to build a minimal Linux system. This system will contain just enough tools to start constructing the final LFS system in Chapter 6 and allow a working environment with more user convenience than a minimum environment would.

There are two steps in building this minimal system. The first step is to build a new and host-independent toolchain (compiler, assembler, linker, libraries, and a few useful utilities). The second step uses this toolchain to build the other essential tools.

The files compiled in this chapter will be installed under the `$LFS/tools` directory to keep them separate from the files installed in the next chapter and the host production directories. Since the packages compiled here are temporary, we do not want them to pollute the soon-to-be LFS system.

5.2. Toolchain Technical Notes

This section explains some of the rationale and technical details behind the overall build method. It is not essential to immediately understand everything in this section. Most of this information will be clearer after performing an actual build. This section can be referred back to at any time during the process.

The overall goal of Chapter 5 is to provide a temporary environment that can be chrooted into and from which can be produced a clean, trouble-free build of the target LFS system in Chapter 6. Along the way, we separate the new system from the host system as much as possible, and in doing so, build a self-contained and self-hosted toolchain. It should be noted that the build process has been designed to minimize the risks for new readers and provide maximum educational value at the same time.

Important

Before continuing, be aware of the name of the working platform, often referred to as the target triplet. Many times, the target triplet will probably be *i686-pc-linux-gnu*. A simple way to determine the name of the target triplet is to run the **config.guess** script that comes with the source for many packages. Unpack the Binutils sources and run the script: **./config.guess** and note the output.

Also be aware of the name of the platform's dynamic linker, often referred to as the dynamic loader (not to be confused with the standard linker **ld** that is part of Binutils). The dynamic linker provided by Glibc finds and loads the shared libraries needed by a program, prepares the program to run, and then runs it. The name of the dynamic linker will usually be `ld-linux.so.2`. On platforms that are less prevalent, the name might be `ld.so.1`, and newer 64 bit platforms might be named something else entirely. The name of the platform's dynamic linker can be determined by looking in the `/lib` directory on the host system. A sure-fire way to determine the name is to inspect a random binary from the host system by running: **readelf -l <name of binary> | grep interpreter** and noting the output. The authoritative reference covering all platforms is in the `shlib-versions` file in the root of the Glibc source tree.

Some key technical points of how the Chapter 5 build method works:

- The process is similar in principle to cross-compiling, whereby tools installed in the same prefix work in cooperation, and thus utilize a little GNU “magic”

- Careful manipulation of the standard linker's library search path ensures programs are linked only against chosen libraries
- Careful manipulation of **gcc**'s `specs` file tells the compiler which target dynamic linker will be used

Binutils is installed first because the **configure** runs of both GCC and Glibc perform various feature tests on the assembler and linker to determine which software features to enable or disable. This is more important than one might first realize. An incorrectly configured GCC or Glibc can result in a subtly broken toolchain, where the impact of such breakage might not show up until near the end of the build of an entire distribution. A test suite failure will usually highlight this error before too much additional work is performed.

Binutils installs its assembler and linker in two locations, `/tools/bin` and `/tools/$TARGET_TRIPLET/bin`. The tools in one location are hard linked to the other. An important facet of the linker is its library search order. Detailed information can be obtained from **ld** by passing it the *`--verbose`* flag. For example, an **`ld --verbose | grep SEARCH`** will illustrate the current search paths and their order. It shows which files are linked by **ld** by compiling a dummy program and passing the *`--verbose`* switch to the linker. For example, **`gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded`** will show all the files successfully opened during the linking.

The next package installed is GCC. An example of what can be seen during its run of **configure** is:

```
checking what assembler to use...
        /tools/i686-pc-linux-gnu/bin/as
checking what linker to use... /tools/i686-pc-linux-gnu/bin/ld
```

This is important for the reasons mentioned above. It also demonstrates that GCC's configure script does not search the PATH directories to find which tools to use. However, during the actual operation of **gcc** itself, the same search paths are not necessarily used. To find out which standard linker **gcc** will use, run: **`gcc -print-prog-name=ld`**.

Detailed information can be obtained from **gcc** by passing it the *`-v`* command line option while compiling a dummy program. For example, **`gcc -v dummy.c`** will show detailed information about the preprocessor, compilation, and assembly stages, including **gcc**'s included search paths and their order.

The next package installed is Glibc. The most important considerations for building Glibc are the compiler, binary tools, and kernel headers. The compiler is generally not an issue since Glibc will always use the **gcc** found in a `PATH` directory. The binary tools and kernel headers can be a bit more complicated. Therefore, take no risks and use the available configure switches to enforce the correct selections. After the run of **configure**, check the contents of the `config.make` file in the `glibc-build` directory for all important details. Note the use of *`CC="gcc -B/tools/bin/"`* to control which binary tools are used and the use of the *`-nostdinc`* and *`-isystem`* flags to control the compiler's include search path. These items highlight an important aspect of the Glibc package—it is very self-sufficient in terms of its build machinery and generally does not rely on toolchain defaults.

After the Glibc installation, make some adjustments to ensure that searching and linking take place only within the `/tools` prefix. Install an adjusted **ld**, which has a hard-wired search path limited to `/tools/lib`. Then amend **gcc**'s specs file to point to the new dynamic linker in `/tools/lib`. This last step is vital to the whole process. As mentioned above, a hard-wired path to a dynamic linker is embedded into every Executable and Link Format (ELF)-shared executable. This can be inspected by running: **`readelf -l <name of binary> | grep interpreter`**. Amending gcc's specs file ensures that every program compiled from here through the end of this chapter will use the new dynamic linker in `/tools/lib`.

For the second pass of GCC, its sources also need to be modified to tell GCC to use the new dynamic linker. Failure to do so will result in the GCC programs themselves having the name of the dynamic linker from the host system's `/lib` directory embedded into them, which would defeat the goal of getting away from the host.

During the second pass of Binutils, we are able to utilize the `--with-lib-path` configure switch to control **ld**'s library search path. From this point onwards, the core toolchain is self-contained and self-hosted. The remainder of the Chapter 5 packages all build against the new Glibc in `/tools`.

Upon entering the chroot environment in Chapter 6, the first major package to be installed is Glibc, due to its self-sufficient nature mentioned above. Once this Glibc is installed into `/usr`, perform a quick changeover of the toolchain defaults, then proceed in building the rest of the target LFS system.

5.3. General Compilation Instructions

When building packages there are several assumptions made within the instructions:

- Several of the packages are patched before compilation, but only when the patch is needed to circumvent a problem. A patch is often needed in both this and the next chapter, but sometimes in only one or the other. Therefore, do not be concerned if instructions for a downloaded patch seem to be missing. Warning messages about *offset* or *fuzz* may also be encountered when applying a patch. Do not worry about these warnings, as the patch was still successfully applied.
- During the compilation of most packages, there will be several warnings that scroll by on the screen. These are normal and can safely be ignored. These warnings are as they appear—warnings about deprecated, but not invalid, use of the C or C++ syntax. C standards change fairly often, and some packages still use the older standard. This is not a problem, but does prompt the warning.

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

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

```
echo $LFS
```

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

- Finally, one last important item must be emphasized:

> **Important**
>
> Before issuing the build instructions for a package, the package should be unpacked as user `lfs`, and a **cd** into the created directory should be performed. The build instructions assume that the **bash** shell is in use.

5.4. Binutils-2.18 - Pass 1

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

Approximate build time: 1 SBU
Required disk space: 213 MB

5.4.1. Installation of Binutils

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

Binutils does not recognize versions of Texinfo newer than 4.9. Fix this issue by applying the following patch:

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

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

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

Note

In order for the SBU values listed in the rest of the book to be of any use, measure the time it takes to build this package from the configuration, up to and including the first install. To achieve this easily, wrap the three commands in a **time** command like this: **`time { ./configure ... && make && make install; }`**.

Now prepare Binutils for compilation:

```
CC="gcc -B/usr/bin/" ../binutils-2.18/configure \
    --prefix=/tools --disable-nls --disable-werror
```

The meaning of the configure options:

`CC="gcc -B/usr/bin/"`
: This forces **gcc** to prefer the linker from the host in `/usr/bin`. This is necessary on some hosts where the new **ld** built here is not compatible with the host's **gcc**.

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

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

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

Continue with compiling the package:

```
make
```

Compilation is now complete. Ordinarily we would now run the test suite, but at this early stage the test suite framework (Tcl, Expect, and DejaGNU) is not yet in place. The benefits of running the tests at this point are minimal since the programs from this first pass will soon be replaced by those from the second.

Install the package:

```
make install
```

Next, prepare the linker for the “Adjusting” phase later on:

```
make -C ld clean
make -C ld LIB_PATH=/tools/lib
cp -v ld/ld-new /tools/bin
```

The meaning of the make parameters:

-C ld clean
: This tells the make program to remove all compiled files in the `ld` subdirectory.

-C ld LIB_PATH=/tools/lib
: This option rebuilds everything in the `ld` subdirectory. Specifying the `LIB_PATH` Makefile variable on the command line allows us to override the default value and point it to the temporary tools location. The value of this variable specifies the linker's default library search path. This preparation is used later in the chapter.

Details on this package are located in Section 6.11.2, “Contents of Binutils.”

5.5. GCC-4.3.2 - Pass 1

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

Approximate build time: 22 SBU
Required disk space: 1.1 GB

5.5.1. Installation of GCC

GCC now requires the GMP and MPFR packages. As these packages may not be included in your host distribution, they will be built with GCC.

```
tar -jxf ../mpfr-2.3.2.tar.bz2
mv mpfr-2.3.2 mpfr
tar -jxf ../gmp-4.2.4.tar.bz2
mv gmp-4.2.4 gmp
```

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

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

Prepare GCC for compilation:

```
CC="gcc -B/usr/bin/" ../gcc-4.3.2/configure \
    --prefix=/tools --with-local-prefix=/tools --disable-nls \
    --disable-shared --disable-libssp --enable-languages=c
```

The meaning of the configure options:

`CC="gcc -B/usr/bin/"`
: This forces **gcc** to prefer the linker from the host in `/usr/bin`. This is necessary on some hosts where the new **ld** built in the previous section is not compatible with the host's **gcc**.

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

`--disable-shared`
: This switch forces GCC to link its internal libraries statically. We do this to avoid possible issues with the host system.

`--disable-libssp`
: This switch prevents a conflict with older versions of glibc which can cause the build to fail.

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

The following command will compile GCC not once, but several times. It uses the programs compiled in a first round to compile itself a second time, and then again a third time. It then compares these second and third compiles to make sure it can reproduce itself flawlessly. This is called “bootstrapping”. Building GCC in this way ensures that it was compiled correctly and is now the default configuration for the released package. Continue with compiling by running:

```
make
```

Compilation is now complete. At this point, the test suite would normally be run, but, as mentioned before, the test suite framework is not in place yet. The benefits of running the tests at this point are minimal since the programs from this first pass will soon be replaced.

Install the package:

```
make install
```

Using *--disable-shared* means that the `libgcc_eh.a` file isn't created and installed. The Glibc package depends on this library as it uses *-lgcc_eh* within its build system. We can satisfy that dependency by creating a symlink to `libgcc.a`, since that file will end up containing the objects normally contained in `libgcc_eh.a`.

```
ln -vs libgcc.a `gcc -print-libgcc-file-name | \
    sed 's/libgcc/&_eh/'`
```

As a finishing touch, create a symlink. Many programs and scripts run **cc** instead of **gcc**, which is used to keep programs generic and therefore usable on all kinds of UNIX systems where the GNU C compiler is not always installed. Running **cc** leaves the system administrator free to decide which C compiler to install:

```
ln -vs gcc /tools/bin/cc
```

Details on this package are located in Section 6.14.2, “Contents of GCC.”

5.6. Linux-2.6.27.4 API Headers

The Linux API Headers expose the kernel's API for use by Glibc.

Approximate build time: 0.1 SBU
Required disk space: 341 MB

5.6.1. Installation of Linux API Headers

The Linux kernel needs to expose an Application Programming Interface (API) for the system's C library (Glibc in LFS) to use. This is done by way of sanitizing various C header files that are shipped in the Linux kernel source tarball.

First, make sure there are no stale files and dependencies lying around from previous activity:

```
make mrproper
```

Now test and extract the user-visible kernel headers from the source. They are placed in an intermediate local directory and copied to the needed location because the extraction process removes any existing files in the target directory.

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

Details on this package are located in Section 6.7.2, “Contents of Linux API Headers.”

5.7. Glibc-2.8-20080929

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

Approximate build time: 7.6 SBU
Required disk space: 407 MB

5.7.1. Installation of Glibc

Fix a potential issue if `/etc/ld.so.preload` is used on the host system.

```
sed -i 's@/etc/ld.so.preload@/tools/etc/ld.so.preload@' elf/rtld.c
```

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

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

Because Glibc no longer supports i386, its developers say to use the compiler flag *-march=i486* when building it for x86 machines. There are several ways to accomplish that, but testing shows that the flag is best placed inside the build variable “CFLAGS”. Instead of overriding completely what Glibc's internal build system uses for CFLAGS, append the new flag to the existing contents of CFLAGS by making use of the special file `configparms`. The -mtune=native flag is also necessary to reset a reasonable value for -mtune that is changed when setting -march.

```
echo "CFLAGS += -march=i486 -mtune=native" > configparms
```

Next, prepare Glibc for compilation:

```
../glibc-2.8-20080929/configure --prefix=/tools \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --with-binutils=/tools/bin \
    --without-gd --with-headers=/tools/include \
    --without-selinux
```

The meaning of the configure options:

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

--enable-add-ons
: This tells Glibc to use the NPTL add-on as its threading library.

--enable-kernel=2.6.0
: This tells Glibc to compile the library with support for 2.6.x Linux kernels.

--with-binutils=/tools/bin
: While not required, this switch ensures that there are no errors pertaining to which Binutils programs get used during the Glibc build.

--without-gd
: This prevents the build of the **memusagestat** program, which insists on linking against the host's libraries (libgd, libpng, libz, etc.).

--with-headers=/tools/include

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

--without-selinux

When building from hosts that include SELinux functionality (e.g., Fedora Core 3), Glibc will build with support for SELinux. As the LFS tools environment does not contain support for SELinux, a Glibc compiled with such support will fail to operate correctly.

During this stage the following warning might appear:

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

The missing or incompatible **msgfmt** program is generally harmless, but it can sometimes cause issues when running the test suite. This **msgfmt** program is part of the Gettext package which the host distribution should provide. If **msgfmt** is present but deemed incompatible, upgrade the host system's Gettext package or continue without it and see if the test suite runs without problems regardless.

Compile the package:

```
make
```

This package does come with a test suite, however, it cannot be run at this time because we do not have a C++ compiler yet.

The install stage of Glibc will issue a harmless warning at the end about the absence of `/tools/etc/ld.so.conf`. Prevent this warning with:

```
mkdir -v /tools/etc
touch /tools/etc/ld.so.conf
```

Install the package:

```
make install
```

Different countries and cultures have varying conventions for how to communicate. These conventions range from the format for representing dates and times to more complex issues, such as the language spoken. The “internationalization” of GNU programs works by locale.

Note

If the test suites are not being run in this chapter (as per the recommendation), there is no need to install the locales now. The appropriate locales will be installed in the next chapter. To install the Glibc locales anyway, use instructions from Section 6.9, “Glibc-2.8-20080929.”

Details on this package are located in Section 6.9.4, “Contents of Glibc.”

5.8. Adjusting the Toolchain

Now that the temporary C libraries have been installed, all tools compiled in the rest of this chapter should be linked against these libraries. In order to accomplish this, the linker and the compiler's specs file need to be adjusted.

The linker, adjusted at the end of the first pass of Binutils, needs to be renamed so that it can be properly found and used. First, backup the original linker, then replace it with the adjusted linker. We'll also create a link to its counterpart in `/tools/$(gcc -dumpmachine)/bin`:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

From this point onwards, everything will link only against the libraries in `/tools/lib`.

The next task is to point GCC to the new dynamic linker. This is done by dumping GCC's “specs” file to a location where GCC will look for it by default. A simple **sed** substitution then alters the dynamic linker that GCC will use.

For the sake of accuracy, it is recommended to use a copy-and-paste method when issuing the following command. Be sure to visually inspect the specs file and verify that all occurrences of “/lib/ld-linux.so.2” have been replaced with “/tools/lib/ld-linux.so.2”:

Important

If working on a platform where the name of the dynamic linker is something other than `ld-linux.so.2`, replace “ld-linux.so.2” with the name of the platform's dynamic linker in the following commands. Refer to Section 5.2, “Toolchain Technical Notes,” if necessary.

```
gcc -dumpspecs | sed 's@/lib/ld-linux.so.2@/tools&@g' \
  > `dirname $(gcc -print-libgcc-file-name)`/specs
```

During the build process, GCC runs a script (**fixincludes**) that scans the system for header files that may need to be fixed (they might contain syntax errors, for example), and installs the fixed versions in a private include directory. There is a possibility that, as a result of this process, some header files from the host system have found their way into GCC's private include directory. As the rest of this chapter only requires the headers from GCC and Glibc, which have both been installed at this point, any “fixed” headers can safely be removed. This helps to avoid any host headers polluting the build environment. Run the following commands to remove the header files in GCC's private include directory (you may find it easier to copy and paste these commands, rather than typing them by hand, due to their length):

```
GCC_FIXED=`dirname $(gcc -print-libgcc-file-name)`/include-fixed &&
find ${GCC_FIXED}/* -maxdepth 0 -xtype d -exec rm -rvf '{}' \; &&
rm -vf `grep -l "DO NOT EDIT THIS FILE" ${GCC_FIXED}/*` &&
unset GCC_FIXED
```

Caution

At this point, it is imperative to stop and ensure that the basic functions (compiling and linking) of the new toolchain are working as expected. To perform a sanity check, run the following commands:

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

If everything is working correctly, there should be no errors, and the output of the last command will be of the form:

```
[Requesting program interpreter:
    /tools/lib/ld-linux.so.2]
```

Note that `/tools/lib` appears as the prefix of the dynamic linker.

If the output is not shown as above or there was no output at all, then something is wrong. Investigate and retrace the steps to find out where the problem is and correct it. This issue must be resolved before continuing on. First, perform the sanity check again, using **gcc** instead of **cc**. If this works, then the `/tools/bin/cc` symlink is missing. Revisit Section 5.5, “GCC-4.3.2 - Pass 1,” and install the symlink. Next, ensure that the `PATH` is correct. This can be checked by running **echo $PATH** and verifying that `/tools/bin` is at the head of the list. If the `PATH` is wrong it could mean that you are not logged in as user `lfs` or that something went wrong back in Section 4.4, “Setting Up the Environment.” Another option is that something may have gone wrong with the specs file amendment above. In this case, redo the specs file amendment, being careful to copy-and-paste the commands.

Once all is well, clean up the test files:

```
rm -v dummy.c a.out
```

Note

Building Tcl in the next section will serve as an additional check that the toolchain has been built properly. If Tcl fails to build, it is an indication that something has gone wrong with the Binutils, GCC, or Glibc installation, but not with Tcl itself.

5.9. Tcl-8.5.5

The Tcl package contains the Tool Command Language.

Approximate build time: 0.5 SBU
Required disk space: 36 MB

5.9.1. Installation of Tcl

This package and the next two (Expect and DejaGNU) are installed to support running the test suites for GCC and Binutils. Installing three packages for testing purposes may seem excessive, but it is very reassuring, if not essential, to know that the most important tools are working properly. Even if the test suites are not run in this chapter (they are not mandatory), these packages are required to run the test suites in Chapter 6.

Prepare Tcl for compilation:

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

Build the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Tcl test suite anyway, issue the following command:

```
TZ=UTC make test
```

The Tcl test suite may experience failures under certain host conditions that are not fully understood. Therefore, test suite failures here are not surprising, and are not considered critical. The `TZ=UTC` parameter sets the time zone to Coordinated Universal Time (UTC), also known as Greenwich Mean Time (GMT), but only for the duration of the test suite run. This ensures that the clock tests are exercised correctly. Details on the `TZ` environment variable are provided in Chapter 7.

Install the package:

```
make install
```

Make the installed library writable so debugging symbols can be removed later:

```
chmod -v u+w /tools/lib/libtcl8.5.so
```

Install Tcl's headers. The next package, Expect, requires them to build.

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

Now make a necessary symbolic link:

```
ln -sv tclsh8.5 /tools/bin/tclsh
```

5.9.2. Contents of Tcl

Installed programs: tclsh (link to tclsh8.5) and tclsh8.5
Installed library: libtcl8.5.so

Short Descriptions

tclsh8.5	The Tcl command shell
tclsh	A link to tclsh8.5
`libtcl8.5.so`	The Tcl library

5.10. Expect-5.43.0

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

Approximate build time: 0.1 SBU
Required disk space: 4 MB

5.10.1. Installation of Expect

First, fix a bug that can result in false failures during the GCC test suite run:

```
patch -Np1 -i ../expect-5.43.0-spawn-1.patch
```

Next, fix a bug that is a result of recent Tcl changes:

```
patch -Np1 -i ../expect-5.43.0-tcl_8.5.5_fix-1.patch
```

Next, force Expect's configure script to use `/bin/stty` instead of a `/usr/local/bin/stty` it may find on the host system. This will ensure that our testsuite tools remain sane for the final builds of our toolchain:

```
cp -v configure{,.orig}
sed 's:/usr/local/bin:/bin:' configure.orig > configure
```

Now prepare Expect for compilation:

```
./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=/tools/include --with-x=no
```

The meaning of the configure options:

`--with-tcl=/tools/lib`
: This ensures that the configure script finds the Tcl installation in the temporary tools location instead of possibly locating an existing one on the host system.

`--with-tclinclude=/tools/include`
: This explicitly tells Expect where to find Tcl's internal headers. Using this option avoids conditions where **configure** fails because it cannot automatically discover the location of Tcl's headers.

`--with-x=no`
: This tells the configure script not to search for Tk (the Tcl GUI component) or the X Window System libraries, both of which may reside on the host system but will not exist in the temporary environment.

Build the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Expect test suite anyway, issue the following command:

```
make test
```

Note that the Expect test suite is known to experience failures under certain host conditions that are not within our control. Therefore, test suite failures here are not surprising and are not considered critical.

Install the package:

```
make SCRIPTS="" install
```

The meaning of the make parameter:

`SCRIPTS=""`
: This prevents installation of the supplementary Expect scripts, which are not needed.

5.10.2. Contents of Expect

Installed program: expect
Installed library: libexpect-5.43.a

Short Descriptions

expect	Communicates with other interactive programs according to a script
`libexpect-5.43.a`	Contains functions that allow Expect to be used as a Tcl extension or to be used directly from C or C++ (without Tcl)

5.11. DejaGNU-1.4.4

The DejaGNU package contains a framework for testing other programs.

Approximate build time: less than 0.1 SBU
Required disk space: 6.2 MB

5.11.1. Installation of DejaGNU

Prepare DejaGNU for compilation:

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

Build and install the package:

```
make install
```

This package does come with a test suite, however, it cannot be run at this time because we do not have a C++ compiler yet.

5.11.2. Contents of DejaGNU

Installed program: runtest

Short Descriptions

runtest A wrapper script that locates the proper **expect** shell and then runs DejaGNU

5.12. GCC-4.3.2 - Pass 2

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

Approximate build time:	6.5 SBU
Required disk space:	865 MB

5.12.1. Re-installation of GCC

The tools required to test GCC and Binutils—Tcl, Expect and DejaGNU—are installed now. GCC and Binutils can now be rebuilt, linking them against the new Glibc and testing them properly (if running the test suites in this chapter). Please note that these test suites are highly dependent on properly functioning PTYs which are provided by the host. PTYs are most commonly implemented via the `devpts` file system. Check to see if the host system is set up correctly in this regard by performing a quick test:

```
expect -c "spawn ls"
```

The response might be:

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

If the above message is received, the host does not have its PTYs set up properly. In this case, there is no point in running the test suites for GCC and Binutils until this issue is resolved. Please consult the LFS FAQ at *http://www.linuxfromscratch.org//lfs/faq.html#no-ptys* for more information on how to get PTYs working.

As previously explained in Section 5.8, "Adjusting the Toolchain", under normal circumstances the GCC **fixincludes** script is run in order to fix potentially broken header files. As GCC-4.3.2 and Glibc-2.8-20080929 have already been installed at this point, and their respective header files are known to not require fixing, the **fixincludes** script is not required. As mentioned previously, the script may in fact pollute the build environment by installing fixed headers from the host system into GCC's private include directory. The running of the **fixincludes** script can be suppressed by issuing the following commands:

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

The bootstrap build performed in Section 5.5, "GCC-4.3.2 - Pass 1" built GCC with the `-fomit-frame-pointer` compiler flag. Non-bootstrap builds omit this flag by default, so apply the following **sed** to use it in order to ensure consistent compiler builds:

```
cp -v gcc/Makefile.in{,.tmp}
sed 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \
  > gcc/Makefile.in
```

The following command will change the location of GCC's default dynamic linker to use the one we installed in `/tools`. It also removes `/usr/include` from GCC's include search path. Doing this now rather than adjusting the specs file after installation ensures that the new dynamic linker is used during the actual build of GCC. That is, all of the binaries created during the build will link against the new Glibc. Issue:

```
for file in $(find gcc/config -name linux64.h -o -name linux.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
  -e 's@/usr@/tools@g' $file.orig > $file
  echo "
#undef STANDARD_INCLUDE_DIR
#define STANDARD_INCLUDE_DIR 0" >> $file
  touch $file.orig
done
```

In case the above seems hard to follow, let's break it down a bit. First we find all the files under the gcc/config directory that are named either `linux.h` or `linux64.h`. For each file found, we copy it to a file of the same name but with an added suffix of “.orig”. Then the first sed expression prepends “/tools” to every instance of “/lib/ld”, “/lib64/ld” or “/lib32/ld”, while the second one replaces hard-coded instances of “/usr”. Then we add our define statements which alter the include search path to the end of the file. Finally, we use **touch** to update the timestamp on the copied files. When used in conjunction with **cp -u**, this prevents unexpected changes to the original files in case the command is inadvertently run twice.

As in the first build of GCC it requires the GMP and MPFR packages. Unpack the tarballs and move them into the required directory names:

```
tar -jxf ../mpfr-2.3.2.tar.bz2
mv mpfr-2.3.2 mpfr
tar -jxf ../gmp-4.2.4.tar.bz2
mv gmp-4.2.4 gmp
```

Create a separate build directory again:

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

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

Now prepare GCC for compilation:

```
../gcc-4.3.2/configure --prefix=/tools \
    --with-local-prefix=/tools --enable-clocale=gnu \
    --enable-shared --enable-threads=posix \
    --enable-__cxa_atexit --enable-languages=c,c++ \
    --disable-libstdcxx-pch --disable-bootstrap
```

The meaning of the new configure options:

`--enable-clocale=gnu`

This option ensures the correct locale model is selected for the C++ libraries under all circumstances. If the configure script finds the *de_DE* locale installed, it will select the correct gnu locale model. However, if the *de_DE* locale is not installed, there is the risk of building Application Binary Interface (ABI)-incompatible C++ libraries because the incorrect generic locale model may be selected.

--enable-threads=posix

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

--enable-__cxa_atexit

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

--enable-languages=c,c++

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

--disable-libstdcxx-pch

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

--disable-bootstrap

Bootstrapping the compiler is now the default for GCC. However, our build method should provide us with a solid compiler without the need to bootstrap each time.

Compile the package:

```
make
```

Compilation is now complete. As previously mentioned, running the test suites for the temporary tools compiled in this chapter is not mandatory. To run the GCC test suite anyway, use the following command:

```
make -k check
```

The *-k* flag is used to make the test suite run through to completion and not stop at the first failure. The GCC test suite is very comprehensive and is almost guaranteed to generate a few failures.

For a discussion of test failures that are of particular importance, please see Section 6.14, “GCC-4.3.2.”

Install the package:

```
make install
```

Caution

At this point, it is imperative to stop and ensure that the basic functions (compiling and linking) of the new toolchain are working as expected. To perform a sanity check, run the following commands:

```
echo 'main(){}' > dummy.c
cc dummy.c
readelf -l a.out | grep ': /tools'
```

If everything is working correctly, there should be no errors, and the output of the last command will be of the form:

```
[Requesting program interpreter:
    /tools/lib/ld-linux.so.2]
```

Note that `/tools/lib` appears as the prefix of the dynamic linker.

If the output is not shown as above or there was no output at all, then something is wrong. Investigate and retrace the steps to find out where the problem is and correct it. This issue must be resolved before continuing on. First, perform the sanity check again, using **gcc** instead of **cc**. If this works, then the `/tools/bin/cc` symlink is missing. Revisit Section 5.5, “GCC-4.3.2 - Pass 1,” and install the symlink. Next, ensure that the `PATH` is correct. This can be checked by running **echo $PATH** and verifying that `/tools/bin` is at the head of the list. If the `PATH` is wrong it could mean that you are not logged in as user `lfs` or that something went wrong back in Section 4.4, “Setting Up the Environment.” Another option is that something may have gone wrong with the specs file amendment above. In this case, redo the specs file amendment, being careful to copy-and-paste the commands.

Once all is well, clean up the test files:

```
rm -v dummy.c a.out
```

Details on this package are located in Section 6.14.2, “Contents of GCC.”

5.13. Binutils-2.18 - Pass 2

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

Approximate build time: 1 SBU
Required disk space: 177 MB

5.13.1. Re-installation of Binutils

Binutils does not recognize versions of Texinfo newer than 4.9. Fix this issue by applying the following patch:

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

Create a separate build directory again:

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

Prepare Binutils for compilation:

```
../binutils-2.18/configure --prefix=/tools \
    --disable-nls --with-lib-path=/tools/lib
```

The meaning of the new configure options:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Binutils test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Now prepare the linker for the “Re-adjusting” phase in the next chapter:

```
make -C ld clean
make -C ld LIB_PATH=/usr/lib:/lib
cp -v ld/ld-new /tools/bin
```

Details on this package are located in Section 6.11.2, “Contents of Binutils.”

5.14. Ncurses-5.6

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

Approximate build time: 0.7 SBU
Required disk space: 30 MB

5.14.1. Installation of Ncurses

Prepare Ncurses for compilation:

```
./configure --prefix=/tools --with-shared \
    --without-debug --without-ada --enable-overwrite
```

The meaning of the configure options:

--without-ada
: This ensures that Ncurses does not build support for the Ada compiler which may be present on the host but will not be available once we enter the **chroot** environment.

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

Compile the package:

```
make
```

This package has a test suite, but it can only be run after the package has been installed. The tests reside in the `test/` directory. See the `README` file in that directory for further details.

Install the package:

```
make install
```

Details on this package are located in Section 6.22.2, “Contents of Ncurses.”

5.15. Bash-3.2

The Bash package contains the Bourne-Again SHell.

Approximate build time: 0.4 SBU
Required disk space: 22 MB

5.15.1. Installation of Bash

Apply fixes for several bugs discovered since the initial release of Bash-3.2:

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

Prepare Bash for compilation:

```
./configure --prefix=/tools --without-bash-malloc \
   ac_cv_func_working_mktime=yes
```

The meaning of the configure options:

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

ac_cv_func_working_mktime=yes
: This parameter bypasses the search for mktime in configure and uses the version in glibc. The is necessary due to a change in gcc that has not been incorporated into this package yet.

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Bash test suite anyway, issue the following command:

```
make tests
```

Install the package:

```
make install
```

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

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

Details on this package are located in Section 6.30.2, “Contents of Bash.”

5.16. Bzip2-1.0.5

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

Approximate build time: less than 0.1 SBU
Required disk space: 4.8 MB

5.16.1. Installation of Bzip2

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

```
make
```

Install the package:

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

Details on this package are located in Section 6.31.2, “Contents of Bzip2.”

5.17. Coreutils-6.12

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

Approximate build time: 0.7 SBU
Required disk space: 83 MB

5.17.1. Installation of Coreutils

There's an internal issue with Coreutils which makes some of the programs behave abnormally if you build using an older kernel. Apply a patch to fix the issue:

```
patch -Np1 -i ../coreutils-6.12-old_build_kernel-1.patch
```

Prepare Coreutils for compilation:

```
./configure --prefix=/tools --enable-install-program=hostname
```

The meaning of the configure options:

`--enable-install-program=hostname`
This enables the **hostname** binary to be built and installed – it is disabled by default but is required by the Perl test suite.

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Coreutils test suite anyway, issue the following command:

```
make RUN_EXPENSIVE_TESTS=yes check
```

The *RUN_EXPENSIVE_TESTS=yes* parameter tells the test suite to run several additional tests that are considered relatively expensive (in terms of CPU power and memory usage) on some platforms, but generally are not a problem on Linux.

Install the package:

```
make install
```

The above command refuses to install `su` because the program cannot be installed setuid root as a non-privileged user. By manually installing it with a different name, we can use it for running tests in the final system as a non-privileged user and we keep a possibly useful **su** from our host first in our PATH. Install it with:

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

Details on this package are located in Section 6.18.2, “Contents of Coreutils.”

5.18. Diffutils-2.8.1

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

Approximate build time: 0.1 SBU
Required disk space: 6.2 MB

5.18.1. Installation of Diffutils

Prepare Diffutils for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Details on this package are located in Section 6.32.2, “Contents of Diffutils.”

5.19. E2fsprogs-1.41.3

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

Approximate build time: 0.4 SBU
Required disk space: 37 MB

5.19.1. Installation of E2fsprogs

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

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

Prepare E2fsprogs for compilation:

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

Compile the package:

```
make
```

Install the static libraries and headers required by Util-linux-ng:

```
make install-libs
```

Make the installed static libraries writable so debugging symbols can be removed later.

```
chmod -v u+w \
    /tools/lib/{libblkid,libcom_err,libe2p,libext2fs,libss,libuuid}.a
```

Details on this package are located in Section 6.17.2, “Contents of E2fsprogs.”

5.20. Findutils-4.4.0

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

Approximate build time:	0.3 SBU
Required disk space:	20 MB

5.20.1. Installation of Findutils

Prepare Findutils for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Findutils test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.35.2, “Contents of Findutils.”

5.21. Gawk-3.1.6

The Gawk package contains programs for manipulating text files.

Approximate build time: 0.3 SBU
Required disk space: 19 MB

5.21.1. Installation of Gawk

Prepare Gawk for compilation:

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

The meaning of the configure option:

ac_cv_func_working_mktime=yes
This parameter bypasses the search for mktime in configure and uses the version in glibc. The is necessary due to a change in gcc that has not been incorporated into this package yet.

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Gawk test suite anyway, issue the following command:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.34.2, “Contents of Gawk.”

5.22. Gettext-0.17

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

Approximate build time: 0.8 SBU
Required disk space: 83 MB

5.22.1. Installation of Gettext

For our temporary set of tools, we only need to build and install one binary from Gettext.

Prepare Gettext for compilation:

```
cd gettext-tools
./configure --prefix=/tools --disable-shared
```

The meaning of the configure option:

--disable-shared
We do not need to install any of the shared Gettext libraries at this time, therefore there is no need to build them.

Compile the package:

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

As only one binary has been compiled, it is not possible to run the testsuite without compiling additional support libraries from the Gettext package. It is therefore not recommended to attempt to run the testsuite at this stage.

Install the **msgfmt** binary:

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

Details on this package are located in Section 6.38.2, “Contents of Gettext.”

5.23. Grep-2.5.3

The Grep package contains programs for searching through files.

Approximate build time: 0.1 SBU
Required disk space: 6.9 MB

5.23.1. Installation of Grep

Prepare Grep for compilation:

```
./configure --prefix=/tools \
    --disable-perl-regexp \
    --without-included-regex
```

The meaning of the configure switches:

--disable-perl-regexp
: This ensures that the **grep** program does not get linked against a Perl Compatible Regular Expression (PCRE) library that may be present on the host but will not be available once we enter the **chroot** environment.

--without-included-regex
: The configure check for glibc's regex library is broken when building against glibc-2.8. This switch forces the use of glibc's regex library.

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Grep test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.39.2, “Contents of Grep.”

5.24. Gzip-1.3.12

The Gzip package contains programs for compressing and decompressing files.

Approximate build time: less than 0.1 SBU
Required disk space: 2.2 MB

5.24.1. Installation of Gzip

The version of the function “futimens” used by Gzip is incompatible with the version that current Glibc provides, so we'll rename the function:

```
for file in gzip.c lib/utimens.{c,h} ; do \
   cp -v $file{,.orig}
   sed 's/futimens/gl_&/' $file.orig > $file
done
```

Prepare Gzip for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Gzip test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.41.2, “Contents of Gzip.”

5.25. M4-1.4.12

The M4 package contains a macro processor.

Approximate build time: 0.2 SBU
Required disk space: 10 MB

5.25.1. Installation of M4

Prepare M4 for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the M4 test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.20.2, “Contents of M4.”

5.26. Make-3.81

The Make package contains a program for compiling packages.

Approximate build time: 0.1 SBU
Required disk space: 9.6 MB

5.26.1. Installation of Make

Prepare Make for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Make test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.46.2, “Contents of Make.”

5.27. Patch-2.5.4

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

Approximate build time: less than 0.1 SBU
Required disk space: 1.6 MB

5.27.1. Installation of Patch

Prepare Patch for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Details on this package are located in Section 6.49.2, “Contents of Patch.”

5.28. Perl-5.10.0

The Perl package contains the Practical Extraction and Report Language.

Approximate build time: 0.9 SBU
Required disk space: 108 MB

5.28.1. Installation of Perl

First apply a series of patches to address security issues and adapt some hard-wired paths to the C library by applying the following patch:

```
patch -Np1 -i ../perl-5.10.0-consolidated-1.patch
```

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

```
sh Configure -des -Dprefix=/tools \
                  -Dstatic_ext='Data/Dumper Fcntl IO POSIX'
```

The meaning of the configure options:

-Dstatic_ext='Data/Dumper Fcntl IO POSIX'
: This tells Perl to build the minimum set of static extensions needed for installing and testing the Coreutils and Glibc packages in the next chapter.

Only a few of the utilities contained in this package, and one of its libraries need to be built:

```
make perl utilities ext/Errno/pm_to_blib
```

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

Install these tools and their libraries:

```
cp -v perl pod/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.10.0
cp -Rv lib/* /tools/lib/perl5/5.10.0
```

Details on this package are located in Section 6.26.2, “Contents of Perl.”

5.29. Sed-4.1.5

The Sed package contains a stream editor.

Approximate build time: 0.1 SBU
Required disk space: 6.1 MB

5.29.1. Installation of Sed

Prepare Sed for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Sed test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.16.2, “Contents of Sed.”

5.30. Tar-1.20

The Tar package contains an archiving program.

Approximate build time: 0.3 SBU
Required disk space: 19.9 MB

5.30.1. Installation of Tar

Prepare Tar for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Tar test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.54.2, “Contents of Tar.”

5.31. Texinfo-4.13a

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

Approximate build time: 0.3 SBU
Required disk space: 20 MB

5.31.1. Installation of Texinfo

Prepare Texinfo for compilation:

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

Compile the package:

```
make
```

Compilation is now complete. As discussed earlier, running the test suite is not mandatory for the temporary tools here in this chapter. To run the Texinfo test suite anyway, issue the following command:

```
make check
```

Install the package:

```
make install
```

Details on this package are located in Section 6.55.2, “Contents of Texinfo.”

5.32. Util-linux-ng-2.14.1

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

Approximate build time: less than 0.1 SBU
Required disk space: 19 MB

5.32.1. Installation of Util-linux-ng

Prepare Util-linux-ng for compilation:

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

Only a few of the utilities contained in this package need to be built:

```
make BLKID_LIBS="-lblkid -luuid" -C mount mount umount
make -C text-utils more
```

The meaning of the make parameter:

BLKID_LIBS="-lblkid -luuid"
When building only a subset of the package, the `libuuid.a` library is not pulled into the build as it is supposed to. This command overrides the default from the `Makefile`.

This package does not come with a test suite.

Copy these programs to the temporary tools directory:

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

Details on this package are located in Section 6.57.3, “Contents of Util-linux-ng.”

5.33. Stripping

The steps in this section are optional, but if the LFS partition is rather small, it is beneficial to learn that unnecessary items can be removed. The executables and libraries built so far contain about 70 MB of unneeded debugging symbols. Remove those symbols with:

```
strip --strip-debug /tools/lib/*
strip --strip-unneeded /tools/{,s}bin/*
```

These commands will skip a number of files, reporting that it does not recognize their file format. Most of these are scripts instead of binaries.

Take care *not* to use `--strip-unneeded` on the libraries. The static ones would be destroyed and the toolchain packages would need to be built all over again.

To save nearly 20 MB more, remove the documentation:

```
rm -rf /tools/{info,man}
```

At this point, you should have at least 850 MB of free space in `$LFS` that can be used to build and install Glibc in the next phase. If you can build and install Glibc, you can build and install the rest too.

5.34. Changing Ownership

Note

The commands in the remainder of this book must be performed while logged in as user `root` and no longer as user `lfs`. Also, double check that `$LFS` is set in `root`'s environment.

Currently, the `$LFS/tools` directory is owned by the user `lfs`, a user that exists only on the host system. If the `$LFS/tools` directory is kept as is, the files are owned by a user ID without a corresponding account. This is dangerous because a user account created later could get this same user ID and would own the `$LFS/tools` directory and all the files therein, thus exposing these files to possible malicious manipulation.

To avoid this issue, you could add the `lfs` user to the new LFS system later when creating the `/etc/passwd` file, taking care to assign it the same user and group IDs as on the host system. Better yet, change the ownership of the `$LFS/tools` directory to user `root` by running the following command:

```
chown -R root:root $LFS/tools
```

Although the `$LFS/tools` directory can be deleted once the LFS system has been finished, it can be retained to build additional LFS systems *of the same book version*. How best to backup `$LFS/tools` is a matter of personal preference and is left as an exercise for the reader.

Caution

If you intend to keep the temporary tools for use in building future LFS systems, *now* is the time to back them up. Subsequent commands in chapter 6 will alter the tools currently in place, rendering them useless for future builds.

Part III. Building the LFS System

Chapter 6. Installing Basic System Software

6.1. Introduction

In this chapter, we enter the building site and start constructing the LFS system in earnest. That is, we chroot into the temporary mini Linux system, make a few final preparations, and then begin installing the packages.

The installation of this software is straightforward. Although in many cases the installation instructions could be made shorter and more generic, we have opted to provide the full instructions for every package to minimize the possibilities for mistakes. The key to learning what makes a Linux system work is to know what each package is used for and why the user (or the system) needs it. For every installed package, a summary of its contents is given, followed by concise descriptions of each program and library the package installed.

If using compiler optimizations, please review the optimization hint at *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*. Compiler optimizations can make a program run slightly faster, but they may also cause compilation difficulties and problems when running the program. If a package refuses to compile when using optimization, try to compile it without optimization and see if that fixes the problem. Even if the package does compile when using optimization, there is the risk it may have been compiled incorrectly because of the complex interactions between the code and build tools. Also note that the `-march` and `-mtune` options may cause problems with the toolchain packages (Binutils, GCC and Glibc). The small potential gains achieved in using compiler optimizations are often outweighed by the risks. First-time builders of LFS are encouraged to build without custom optimizations. The subsequent system will still run very fast and be stable at the same time.

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

Before the installation instructions, each installation page provides information about the package, including a concise description of what it contains, approximately how long it will take to build, and how much disk space is required during this building process. Following the installation instructions, there is a list of programs and libraries (along with brief descriptions of these) that the package installs.

6.2. Preparing Virtual Kernel File Systems

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

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

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

6.2.1. Creating Initial Device Nodes

When the kernel boots the system, it requires the presence of a few device nodes, in particular the `console` and `null` devices. The device nodes will be created on the hard disk so that they are available before **udevd** has been started, and additionally when Linux is started with *`init=/bin/bash`*. Create the devices by running the following commands:

```
mknod -m 600 $LFS/dev/console c 5 1
mknod -m 666 $LFS/dev/null c 1 3
```

6.2.2. Mounting and Populating /dev

The recommended method of populating the `/dev` directory with devices is to mount a virtual filesystem (such as `tmpfs`) on the `/dev` directory, and allow the devices to be created dynamically on that virtual filesystem as they are detected or accessed. This is generally done during the boot process by Udev. Since this new system does not yet have Udev and has not yet been booted, it is necessary to mount and populate `/dev` manually. This is accomplished by bind mounting the host system's `/dev` directory. A bind mount is a special type of mount that allows you to create a mirror of a directory or mount point to some other location. Use the following command to achieve this:

```
mount -v --bind /dev $LFS/dev
```

6.2.3. Mounting Virtual Kernel File Systems

Now mount the remaining virtual kernel filesystems:

```
mount -vt devpts devpts $LFS/dev/pts
mount -vt tmpfs shm $LFS/dev/shm
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
```

6.3. Package Management

Package Management is an often requested addition to the LFS Book. A Package Manager allows tracking the installation of files making it easy to remove and upgrade packages. As well as the binary and library files, a package manager will handle the installation of configuration files. Before you begin to wonder, NO—this section will not talk about nor recommend any particular package manager. What it provides is a roundup of the more popular techniques and how they work. The perfect package manager for you may be among these techniques or may be a combination of two or more of these techniques. This section briefly mentions issues that may arise when upgrading packages.

Some reasons why no package manager is mentioned in LFS or BLFS include:

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

There are some hints written on the topic of package management. Visit the *Hints Project* and see if one of them fits your need.

6.3.1. Upgrade Issues

A Package Manager makes it easy to upgrade to newer versions when they are released. Generally the instructions in the LFS and BLFS Book can be used to upgrade to the newer versions. Here are some points that you should be aware of when upgrading packages, especially on a running system.

- If one of the toolchain packages (Glibc, GCC or Binutils) needs to be upgraded to a newer minor version, it is safer to rebuild LFS. Though you *may* be able to get by rebuilding all the packages in their dependency order, we do not recommend it. For example, if glibc-2.2.x needs to be updated to glibc-2.3.x, it is safer to rebuild. For micro version updates, a simple reinstallation usually works, but is not guaranteed. For example, upgrading from glibc-2.3.4 to glibc-2.3.5 will not usually cause any problems.

- If a package containing a shared library is updated, and if the name of the library changes, then all the packages dynamically linked to the library need to be recompiled to link against the newer library. (Note that there is no correlation between the package version and the name of the library.) For example, consider a package foo-1.2.3 that installs a shared library with name `libfoo.so.1`. Say you upgrade the package to a newer version foo-1.2.4 that installs a shared library with name `libfoo.so.2`. In this case, all packages that are dynamically linked to `libfoo.so.1` need to be recompiled to link against `libfoo.so.2`. Note that you should not remove the previous libraries until the dependent packages are recompiled.

6.3.2. Package Management Techniques

The following are some common package management techniques. Before making a decision on a package manager, do some research on the various techniques, particularly the drawbacks of the particular scheme.

6.3.2.1. It is All in My Head!

Yes, this is a package management technique. Some folks do not find the need for a package manager because they know the packages intimately and know what files are installed by each package. Some users also do not need any package management because they plan on rebuilding the entire system when a package is changed.

6.3.2.2. Install in Separate Directories

This is a simplistic package management that does not need any extra package to manage the installations. Each package is installed in a separate directory. For example, package foo-1.1 is installed in `/usr/pkg/foo-1.1` and a symlink is made from `/usr/pkg/foo` to `/usr/pkg/foo-1.1`. When installing a new version foo-1.2, it is installed in `/usr/pkg/foo-1.2` and the previous symlink is replaced by a symlink to the new version.

Environment variables such as `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` and `CPPFLAGS` need to be expanded to include `/usr/pkg/foo`. For more than a few packages, this scheme becomes unmanageable.

6.3.2.3. Symlink Style Package Management

This is a variation of the previous package management technique. Each package is installed similar to the previous scheme. But instead of making the symlink, each file is symlinked into the `/usr` hierarchy. This removes the need to expand the environment variables. Though the symlinks can be created by the user to automate the creation, many package managers have been written using this approach. A few of the popular ones include Stow, Epkg, Graft, and Depot.

The installation needs to be faked, so that the package thinks that it is installed in `/usr` though in reality it is installed in the `/usr/pkg` hierarchy. Installing in this manner is not usually a trivial task. For example, consider that you are installing a package libfoo-1.1. The following instructions may not install the package properly:

```
./configure --prefix=/usr/pkg/libfoo/1.1
make
make install
```

The installation will work, but the dependent packages may not link to libfoo as you would expect. If you compile a package that links against libfoo, you may notice that it is linked to `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` instead of `/usr/lib/libfoo.so.1` as you would expect. The correct approach is to use the `DESTDIR` strategy to fake installation of the package. This approach works as follows:

```
./configure --prefix=/usr
make
make DESTDIR=/usr/pkg/libfoo/1.1 install
```

Most packages support this approach, but there are some which do not. For the non-compliant packages, you may either need to manually install the package, or you may find that it is easier to install some problematic packages into `/opt`.

6.3.2.4. Timestamp Based

In this technique, a file is timestamped before the installation of the package. After the installation, a simple use of the **find** command with the appropriate options can generate a log of all the files installed after the timestamp file was created. A package manager written with this approach is install-log.

Though this scheme has the advantage of being simple, it has two drawbacks. If, during installation, the files are installed with any timestamp other than the current time, those files will not be tracked by the package manager. Also, this scheme can only be used when one package is installed at a time. The logs are not reliable if two packages are being installed on two different consoles.

6.3.2.5. Tracing Installation Scripts

In this approach, the commands that the installation scripts perform are recorded. There are two techniques that one can use:

The `LD_PRELOAD` environment variable can be set to point to a library to be preloaded before installation. During installation, this library tracks the packages that are being installed by attaching itself to various executables such as **cp**, **install**, **mv** and tracking the system calls that modify the filesystem. For this approach to work, all the executables need to be dynamically linked without the suid or sgid bit. Preloading the library may cause some unwanted side-effects during installation. Therefore, it is advised that one performs some tests to ensure that the package manager does not break anything and logs all the appropriate files.

The second technique is to use **strace**, which logs all system calls made during the execution of the installation scripts.

6.3.2.6. Creating Package Archives

In this scheme, the package installation is faked into a separate tree as described in the Symlink style package management. After the installation, a package archive is created using the installed files. This archive is then used to install the package either on the local machine or can even be used to install the package on other machines.

This approach is used by most of the package managers found in the commercial distributions. Examples of package managers that follow this approach are RPM (which, incidentally, is required by the *Linux Standard Base Specification*), pkg-utils, Debian's apt, and Gentoo's Portage system. A hint describing how to adopt this style of package management for LFS systems is located at *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

6.3.2.7. User Based Management

This scheme, unique to LFS, was devised by Matthias Benkmann, and is available from the *Hints Project*. In this scheme, each package is installed as a separate user into the standard locations. Files belonging to a package are easily identified by checking the user ID. The features and shortcomings of this approach are too complex to describe in this section. For the details please see the hint at *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

6.4. Entering the Chroot Environment

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

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

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

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

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

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

Note

It is important that all the commands throughout the remainder of this chapter and the following chapters are run from within the chroot environment. If you leave this environment for any reason (rebooting for example), ensure that the virtual kernel filesystems are mounted as explained in Section 6.2.2, “Mounting and Populating /dev” and Section 6.2.3, “Mounting Virtual Kernel File Systems” and enter chroot again before continuing with the installation.

6.5. Creating Directories

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

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

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

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

6.5.1. FHS Compliance Note

The directory tree is based on the Filesystem Hierarchy Standard (FHS) (available at *http://www.pathname.com/fhs/*). In addition to the FHS, we create compatibility symlinks for the man, doc, and info directories since many packages still try to install their documentation into /usr/<directory> or /usr/local/<directory> as opposed to /usr/share/<directory> or /usr/local/share/<directory>. The FHS also stipulates the existence of /usr/local/games and /usr/share/games. The FHS is not precise as to the structure of the /usr/local/share subdirectory, so we create only the directories that are needed. However, feel free to create these directories if you prefer to conform more strictly to the FHS.

6.6. Creating Essential Files and Symlinks

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

```
ln -sv /tools/bin/{bash,cat,echo,grep,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{,.6} /usr/lib
ln -sv bash /bin/sh
```

A proper Linux system maintains a list of the mounted file systems in the file /etc/mtab. Normally, this file would be created when we mount a new file system. Since we will not be mounting any file systems inside our chroot environment, create an empty file for utilities that expect the presence of /etc/mtab:

```
touch /etc/mtab
```

In order for user root to be able to login and for the name “root” to be recognized, there must be relevant entries in the /etc/passwd and /etc/group files.

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

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

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

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

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
uucp:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
mail:x:34:
nogroup:x:99:
EOF
```

The created groups are not part of any standard—they are groups decided on in part by the requirements of the Udev configuration in this chapter, and in part by common convention employed by a number of existing Linux distributions. The Linux Standard Base (LSB, available at *http://www.linuxbase.org*) recommends only that, besides the group `root` with a Group ID (GID) of 0, a group `bin` with a GID of 1 be present. All other group names and GIDs can be chosen freely by the system administrator since well-written programs do not depend on GID numbers, but rather use the group's name.

To remove the “I have no name!” prompt, start a new shell. Since a full Glibc was installed in Chapter 5 and the `/etc/passwd` and `/etc/group` files have been created, user name and group name resolution will now work:

```
exec /tools/bin/bash --login +h
```

Note the use of the *+h* directive. This tells **bash** not to use its internal path hashing. Without this directive, **bash** would remember the paths to binaries it has executed. To ensure the use of the newly compiled binaries as soon as they are installed, the *+h* directive will be used for the duration of this chapter.

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

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

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

6.7. Linux-2.6.27.4 API Headers

The Linux API Headers expose the kernel's API for use by Glibc.

Approximate build time: 0.1 SBU
Required disk space: 341 MB

6.7.1. Installation of Linux API Headers

The Linux kernel needs to expose an Application Programming Interface (API) for the system's C library (Glibc in LFS) to use. This is done by way of sanitizing various C header files that are shipped in the Linux kernel source tarball.

First, make sure there are no stale files and dependencies lying around from previous activity:

```
make mrproper
```

Now test and extract the user-visible kernel headers from the source. They are placed in an intermediate local directory and copied to the needed location because the extraction process removes any existing files in the target directory.

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

6.7.2. Contents of Linux API Headers

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

Short Descriptions

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

6.8. Man-pages-3.11

The Man-pages package contains over 1,900 man pages.

Approximate build time: less than 0.1 SBU
Required disk space: 21 MB

6.8.1. Installation of Man-pages

Install Man-pages by running:

```
make install
```

6.8.2. Contents of Man-pages

Installed files: various man pages

Short Descriptions

`man pages` Describe C programming language functions, important device files, and significant configuration files

6.9. Glibc-2.8-20080929

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

Approximate build time:	17.7 SBU testsuite included
Required disk space:	801 MB testsuite included

6.9.1. Installation of Glibc

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

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

In the vi_VN.TCVN locale, **bash** enters an infinite loop at startup. It is unknown whether this is a **bash** bug or a Glibc problem. Disable installation of this locale in order to avoid the problem:

```
sed -i '/vi_VN.TCVN/d' localedata/SUPPORTED
```

First apply two patches which correct failures in the test suite:

```
patch -Np1 -i ../glibc-2.8-20080929-iconv_tests-1.patch
patch -Np1 -i ../glibc-2.8-20080929-ildoubl_test-1.patch
```

The **ldd** shell script contains Bash-specific syntax. Change its default program interpreter to **/bin/bash** in case another **/bin/sh** is installed as described in the *Shells* chapter of the BLFS book:

```
sed -i 's|@BASH@|/bin/bash|' elf/ldd.bash.in
```

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

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

Again, add the needed compiler flag to CFLAGS:

```
echo "CFLAGS += -march=i486 -mtune=native" > configparms
```

Prepare Glibc for compilation:

```
../glibc-2.8-20080929/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

The meaning of the new configure options:

--libexecdir=/usr/lib/glibc
: This changes the location of the **pt_chown** program from its default of `/usr/libexec` to `/usr/lib/glibc`.

Compile the package:

```
make
```

Important

In this section, the test suite for Glibc is considered critical. Do not skip it under any circumstance.

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

```
cp -v ../glibc-2.8-20080929/iconvdata/gconv-modules iconvdata
make -k check 2>&1 | tee glibc-check-log
grep Error glibc-check-log
```

You will probably see an expected (ignored) failure in the *posix/annexc* test. In addition the Glibc test suite is somewhat dependent on the host system. This is a list of the most common issues:

- The *nptl/tst-cancel1* test will fail when using the 4.1 series of GCC.
- The *nptl/tst-clock2* and *tst-attr3* tests sometimes fail. The reason is not completely understood, but indications are that a heavy system load can trigger these failures.
- The math tests sometimes fail when running on systems where the CPU is not a relatively new genuine Intel or authentic AMD processor.
- If you have mounted the LFS partition with the *`noatime`* option, the *atime* test will fail. As mentioned in Section 2.4, “Mounting the New Partition”, do not use the *`noatime`* option while building LFS.
- When running on older and slower hardware or on systems under load, some tests can fail because of test timeouts being exceeded.

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

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

Install the package:

```
make install
```

The locales that can make the system respond in a different language were not installed by the above command. None of the locales are required, but if some of them are missing, testsuites of the future packages would skip important testcases.

Individual locales can be installed using the **localedef** program. E.g., the first **localedef** command below combines the `/usr/share/i18n/locales/de_DE` charset-independent locale definition with the `/usr/share/i18n/charmaps/ISO-8859-1.gz` charmap definition and appends the result to the `/usr/lib/locale/locale-archive` file. The following instructions will install the minimum set of locales necessary for the optimal coverage of tests:

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

In addition, install the locale for your own country, language and character set.

Alternatively, install all locales listed in the `glibc-2.8-20080929/localedata/SUPPORTED` file (it includes every locale listed above and many more) at once with the following time-consuming command:

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

Then use the **localedef** command to create and install locales not listed in the `glibc-2.8-20080929/localedata/SUPPORTED` file in the unlikely case you need them.

6.9.2. Configuring Glibc

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

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

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

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

One way to determine the local time zone, run the following script:

```
tzselect
```

After answering a few questions about the location, the script will output the name of the time zone (e.g., *America/Edmonton*). There are also some other possible timezones listed in `/usr/share/zoneinfo` such as *Canada/Eastern* or *EST5EDT* that are not identified by the script but can be used.

Then create the `/etc/localtime` file by running:

```
cp -v --remove-destination /usr/share/zoneinfo/<xxx> \
    /etc/localtime
```

Replace `<xxx>` with the name of the time zone selected (e.g., Canada/Eastern).

The meaning of the cp option:

`--remove-destination`

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

6.9.3. Configuring the Dynamic Loader

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

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

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

/usr/local/lib
/opt/lib

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

6.9.4. Contents of Glibc

Installed programs:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump, and zic
Installed libraries:	ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so, and libutil.{a,so}

Short Descriptions

catchsegv	Can be used to create a stack trace when a program terminates with a segmentation fault
gencat	Generates message catalogues
getconf	Displays the system configuration values for file system specific variables
getent	Gets entries from an administrative database
iconv	Performs character set conversion
iconvconfig	Creates fastloading **iconv** module configuration files
ldconfig	Configures the dynamic linker runtime bindings
ldd	Reports which shared libraries are required by each given program or shared library
lddlibc4	Assists **ldd** with object files
locale	Prints various information about the current locale
localedef	Compiles locale specifications
mtrace	Reads and interprets a memory trace file and displays a summary in human-readable format
nscd	A daemon that provides a cache for the most common name service requests
pcprofiledump	Dumps information generated by PC profiling
pt_chown	A helper program for **grantpt** to set the owner, group and access permissions of a slave pseudo terminal
rpcgen	Generates C code to implement the Remote Procedure Call (RPC) protocol
rpcinfo	Makes an RPC call to an RPC server

sln	A statically linked **ln** program
sprof	Reads and displays shared object profiling data
tzselect	Asks the user about the location of the system and reports the corresponding time zone description
xtrace	Traces the execution of a program by printing the currently executed function
zdump	The time zone dumper
zic	The time zone compiler
`ld.so`	The helper program for shared library executables
`libBrokenLocale`	Used internally by Glibc as a gross hack to get broken programs (e.g., some Motif applications) running. See comments in `glibc-2.8-20080929/locale/broken_cur_max.c` for more information
`libSegFault`	The segmentation fault signal handler, used by **catchsegv**
`libanl`	An asynchronous name lookup library
`libbsd-compat`	Provides the portability needed in order to run certain Berkeley Software Distribution (BSD) programs under Linux
`libc`	The main C library
`libcidn`	Used internally by Glibc for handling internationalized domain names in the `getaddrinfo()` function
`libcrypt`	The cryptography library
`libdl`	The dynamic linking interface library
`libg`	Dummy library containing no functions. Previously was a runtime library for **g++**
`libieee`	Linking in this module forces error handling rules for math functions as defined by the Institute of Electrical and Electronic Engineers (IEEE). The default is POSIX.1 error handling
`libm`	The mathematical library
`libmcheck`	Turns on memory allocation checking when linked to
`libmemusage`	Used by **memusage** to help collect information about the memory usage of a program
`libnsl`	The network services library
`libnss`	The Name Service Switch libraries, containing functions for resolving host names, user names, group names, aliases, services, protocols, etc.
`libpcprofile`	Contains profiling functions used to track the amount of CPU time spent in specific source code lines
`libpthread`	The POSIX threads library
`libresolv`	Contains functions for creating, sending, and interpreting packets to the Internet domain name servers
`librpcsvc`	Contains functions providing miscellaneous RPC services
`librt`	Contains functions providing most of the interfaces specified by the POSIX.1b Realtime Extension
`libthread_db`	Contains functions useful for building debuggers for multi-threaded programs

`libutil`	Contains code for “standard” functions used in many different Unix utilities

6.10. Re-adjusting the Toolchain

Now that the final C libraries have been installed, it is time to adjust the toolchain again. The toolchain will be adjusted so that it will link any newly compiled program against these new libraries. This is a similar process used in the “Adjusting” phase in the beginning of Chapter 5, but with the adjustments reversed. In Chapter 5, the chain was guided from the host's `/{,usr/}lib` directories to the new `/tools/lib` directory. Now, the chain will be guided from that same `/tools/lib` directory to the LFS `/{,usr/}lib` directories.

First, backup the `/tools` linker, and replace it with the adjusted linker we made in chapter 5. We'll also create a link to its counterpart in `/tools/$(gcc -dumpmachine)/bin`:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

Next, amend the GCC specs file so that it points to the new dynamic linker, and so that GCC knows where to find the correct headers and Glibc start files. A **sed** command accomplishes this:

> **Important**
>
> If working on a platform where the name of the dynamic linker is something other than `ld-linux.so.2`, substitute “ld-linux.so.2” with the name of the platform's dynamic linker in the following commands. Refer to Section 5.2, “Toolchain Technical Notes,” if necessary.

```
gcc -dumpspecs | sed \
    -e 's@/tools/lib/ld-linux.so.2@/lib/ld-linux.so.2@g' \
    -e '/\*startfile_prefix_spec:/{n;s@.*@/usr/lib/ @}' \
    -e '/\*cpp:/{n;s@$@ -isystem /usr/include@}' > \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

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

It is imperative at this point to ensure that the basic functions (compiling and linking) of the adjusted toolchain are working as expected. To do this, perform the following sanity checks:

```
echo 'main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

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

Now make sure that we're setup to use the correct startfiles:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
/usr/lib/crt1.o succeeded
/usr/lib/crti.o succeeded
/usr/lib/crtn.o succeeded
```

Verify that the compiler is searching for the correct header files:

```
grep -B1 '^ /usr/include' dummy.log
```

This command should return successfully with the following output:

```
#include <...> search starts here:
 /usr/include
```

Next, verify that the new linker is being used with the correct search paths:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
SEARCH_DIR("/tools/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
```

Next make sure that we're using the correct libc:

```
grep "/lib/libc.so.6 " dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
attempt to open /lib/libc.so.6 succeeded
```

Lastly, make sure GCC is using the correct dynamic linker:

```
grep found dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name):

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

If the output does not appear as shown above or is not received at all, then something is seriously wrong. Investigate and retrace the steps to find out where the problem is and correct it. The most likely reason is that something went wrong with the specs file adjustment. Any issues will need to be resolved before continuing on with the process.

Once everything is working correctly, clean up the test files:

```
rm -v dummy.c a.out dummy.log
```

6.11. Binutils-2.18

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

Approximate build time:	1.7 SBU testsuite included
Required disk space:	186 MB testsuite included

6.11.1. Installation of Binutils

Verify that the PTYs are working properly inside the chroot environment. Check that everything is set up correctly by performing a simple test:

```
expect -c "spawn ls"
```

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

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

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

Binutils does not recognize versions of Texinfo newer than 4.9. Fix this issue by applying the following patch:

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

Apply the following patch to prevent some failures when running the the test suite:

```
patch -Np1 -i ../binutils-2.18-GCC43-1.patch
```

Suppress the installation of an outdated `standards.info` file as a newer one is installed later on in the Autoconf instructions:

```
rm -fv etc/standards.info
sed -i.bak '/^INFO/s/standards.info //' etc/Makefile.in
```

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

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

Prepare Binutils for compilation:

```
../binutils-2.18/configure --prefix=/usr \
    --enable-shared
```

Compile the package:

```
make tooldir=/usr
```

The meaning of the make parameter:

`tooldir=/usr`

Normally, the tooldir (the directory where the executables will ultimately be located) is set to `$(exec_prefix)/$(target_alias)`. For example, i686 machines would expand that to

/usr/i686-pc-linux-gnu. Because this is a custom system, this target-specific directory in /usr is not required. $(exec_prefix)/$(target_alias) would be used if the system was used to cross-compile (for example, compiling a package on an Intel machine that generates code that can be executed on PowerPC machines).

Important

The test suite for Binutils in this section is considered critical. Do not skip it under any circumstances.

Test the results:

```
make check
```

Install the package:

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

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

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

6.11.2. Contents of Binutils

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

Installed libraries: libiberty.a, libbfd.{a,so}, and libopcodes.{a,so}

Short Descriptions

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

ar Creates, modifies, and extracts from archives

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

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

gprof Displays call graph profile data

ld A linker that combines a number of object and archive files into a single file, relocating their data and tying up symbol references

nm Lists the symbols occurring in a given object file

objcopy Translates one type of object file into another

objdump Displays information about the given object file, with options controlling the particular information to display; the information shown is useful to programmers who are working on the compilation tools

ranlib Generates an index of the contents of an archive and stores it in the archive; the index lists all of the symbols defined by archive members that are relocatable object files

readelf Displays information about ELF type binaries

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

6.12. GMP-4.2.4

The GMP package contains math libraries. These have useful functions for arbitrary precision arithmetic.

Approximate build time:	1.5 SBU testsuite included
Required disk space:	39.4 MB testsuite included

6.12.1. Installation of GMP

Prepare GMP for compilation:

```
./configure --prefix=/usr --enable-cxx --enable-mpbsd
```

The meaning of the new configure option:

--enable-cxx
  This parameter enables C++ support

Compile the package:

```
make
```

> **Important**
>
> The test suite for GMP in this section is considered critical. Do not skip it under any circumstances.

Test the results:

```
make check 2>&1 | tee gmp-check-log
```

Ensure that all 139 tests in the test suite ran successfully by issuing the following command:

```
awk '/tests passed/{total+=$2} ; END{print total}' gmp-check-log
```

Install the package:

```
make install
```

If desired, install the documentation:

```
mkdir -v /usr/share/doc/gmp-4.2.4
cp    -v doc/{isa_abi_headache,configuration} doc/*.html \
         /usr/share/doc/gmp-4.2.4
```

6.12.2. Contents of GMP

Installed Libraries:	libgmp.{a,so}, libgmpxx.{a,so}, and libmp.{a,so}

Short Descriptions

libgmp	Contains precision math functions.
libgmpxx	Contains C++ precision math functions.
libmp	Contains the Berkeley MP math functions.

6.13. MPFR-2.3.2

The MPFR package contains functions for multiple precision math.

Approximate build time:	1.2 SBU testsuite included
Required disk space:	39.4 MB testsuite included

6.13.1. Installation of MPFR

Prepare MPFR for compilation:

```
./configure --prefix=/usr --enable-thread-safe
```

Compile the package:

```
make
```

Important

The test suite for MPFR in this section is considered critical. Do not skip it under any circumstances.

Test the results and ensure that all 134 tests passed:

```
make check
```

Install the package:

```
make install
```

6.13.2. Contents of MPFR

Installed Libraries:	mpfr.so

Short Descriptions

mpfr	Contains multiple-precision math functions.

6.14. GCC-4.3.2

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

Approximate build time: 25 SBU testsuite included
Required disk space: 1.1 GB testsuite included

6.14.1. Installation of GCC

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

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

The bootstrap build performed in Section 5.5, "GCC-4.3.2 - Pass 1" built GCC with the `-fomit-frame-pointer` compiler flag. Non-bootstrap builds omit this flag by default, so apply the following **sed** to use it in order to ensure consistent compiler builds:

```
sed -i 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in
```

The **fixincludes** script is known to occasionally erroneously attempt to "fix" the system headers installed so far. As the headers installed by GCC-4.3.2 and Glibc-2.8-20080929 are known to not require fixing, issue the following command to prevent the **fixincludes** script from running:

```
sed -i 's@\./fixinc\.sh@-c true@' gcc/Makefile.in
```

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

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

Prepare GCC for compilation:

```
../gcc-4.3.2/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-clocale=gnu --enable-languages=c,c++ \
    --disable-bootstrap
```

Note that for other languages, there are some prerequsites that are not available. See the BLFS Book for instructions on how to build all the GCC supported languages.

Compile the package:

```
make
```

Important

In this section, the test suite for GCC is considered critical. Do not skip it under any circumstance.

Test the results, but do not stop at errors:

```
make -k check
```

To receive a summary of the test suite results, run:

```
../gcc-4.3.2/contrib/test_summary
```

For only the summaries, pipe the output through **`grep -A7 Summ`**.

Results can be compared with those located at *http://www.linuxfromscratch.org/lfs/build-logs/6.4/*.

A few unexpected failures cannot always be avoided. The GCC developers are usually aware of these issues, but have not resolved them yet. In particular, the `libmudflap` tests are known be particularly problematic as a result of a bug in GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). Unless the test results are vastly different from those at the above URL, it is safe to continue.

Install the package:

```
make install
```

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

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

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

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

Now that our final toolchain is in place, it is important to again ensure that compiling and linking will work as expected. We do this by performing the same sanity checks as we did earlier in the chapter:

```
echo 'main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Now make sure that we're setup to use the correct startfiles:

```
grep -o '/usr/lib.*/crt[1in].*succeeded' dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.3.2/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.3.2/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.3.2/../../../crtn.o succeeded
```

Verify that the compiler is searching for the correct header files:

```
grep -B4 '^ /usr/include' dummy.log
```

This command should return successfully with the following output:

```
#include <...> search starts here:
 /usr/local/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.3.2/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.3.2/include-fixed
 /usr/include
```

> **Note**
>
> As of version 4.3.0, GCC now unconditionally installs the `limits.h` file into the private `include-fixed` directory, and that directory is required to be in place.

Next, verify that the new linker is being used with the correct search paths:

```
grep 'SEARCH.*/usr/lib' dummy.log |sed 's|; |\n|g'
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Next make sure that we're using the correct libc:

```
grep "/lib/libc.so.6 " dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be:

```
attempt to open /lib/libc.so.6 succeeded
```

Lastly, make sure GCC is using the correct dynamic linker:

```
grep found dummy.log
```

If everything is working correctly, there should be no errors, and the output of the last command will be (allowing for platform-specific differences in dynamic linker name):

```
found ld-linux.so.2 at /lib/ld-linux.so.2
```

If the output does not appear as shown above or is not received at all, then something is seriously wrong. Investigate and retrace the steps to find out where the problem is and correct it. The most likely reason is that something went wrong with the specs file adjustment. Any issues will need to be resolved before continuing on with the process.

Once everything is working correctly, clean up the test files:

```
rm -v dummy.c a.out dummy.log
```

6.14.2. Contents of GCC

Installed programs:	c++, cc (link to gcc), cpp, g++, gcc, gccbug, and gcov
Installed libraries:	libgcc.a, libgcc_eh.a, libgcc_s.so, libmudflap.{a,so}, libssp.{a,so}, libstdc++.{a,so}, and libsupc++.a

Short Descriptions

c++	The C++ compiler
cc	The C compiler
cpp	The C preprocessor; it is used by the compiler to expand the #include, #define, and similar statements in the source files

g++	The C++ compiler
gcc	The C compiler
gccbug	A shell script used to help create useful bug reports
gcov	A coverage testing tool; it is used to analyze programs to determine where optimizations will have the most effect
`libgcc`	Contains run-time support for **gcc**
`libmudflap`	Contains routines that support GCC's bounds checking functionality
`libssp`	Contains routines supporting GCC's stack-smashing protection functionality
`libstdc++`	The standard C++ library
`libsupc++`	Provides supporting routines for the C++ programming language

6.15. Berkeley DB-4.7.25

The Berkeley DB package contains programs and utilities used by many other applications for database related functions.

Approximate build time: 1.9 SBU
Required disk space: 120 MB

Other Installation Possibilities

There are instructions to build this package in the BLFS book if you need to build the RPC server or additional language bindings. The additional language bindings will require additional packages to be installed. See *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db* for suggested installation instructions.

Also, GDBM *could* be used in place of Berkeley DB to satisfy Man-DB. However, since Berkeley DB is considered a core part of the LFS build, it will not be listed as a dependency for any package in the BLFS book. Likewise, many hours go into testing LFS with Berkeley DB installed, not with GDBM. If you fully understand the risks versus benefits of using GDBM and wish to use it anyway, see the BLFS instructions located at *http://www.linuxfromscratch.org/blfs/view/svn/general/gdbm.html*

6.15.1. Installation of Berkeley DB

Apply an upstream patch so that replication clients can open a sequence:

```
patch -Np1 -i ../db-4.7.25-upstream_fixes-1.patch
```

Prepare Berkeley DB for compilation:

```
cd build_unix
../dist/configure --prefix=/usr --enable-compat185 --enable-cxx
```

The meaning of the configure options:

--enable-compat185
This option enables building Berkeley DB 1.85 compatibility API.

--enable-cxx
This option enables building C++ API libraries.

Compile the package:

```
make
```

It is not possible to test the package meaningfully, because that would involve building TCL bindings. TCL bindings cannot be built properly now because TCL is linked against Glibc in `/tools`, not against Glibc in `/usr`.

Install the package:

```
make docdir=/usr/share/doc/db-4.7.25 install
```

The meaning of the make parameter:

docdir=...
This variable specifies the correct place for the documentation.

Fix the ownership of the installed documentation:

```
chown -Rv root:root /usr/share/doc/db-4.7.25
```

6.15.2. Contents of Berkeley DB

Installed programs: db_archive, db_checkpoint, db_deadlock, db_dump, db_hotbackup, db_load, db_printlog, db_recover, db_stat, db_upgrade, and db_verify
Installed libraries: libdb.{so,ar}and libdb_cxx.r{o,ar}

Short Descriptions

db_archive	Prints the pathnames of log files that are no longer in use
db_checkpoint	A daemon used to monitor and checkpoint database logs
db_deadlock	A daemon used to abort lock requests when deadlocks are detected
db_dump	Converts database files to a plain-text file format readable by **db_load**
db_hotbackup	Creates “hot backup” or “hot failover” snapshots of Berkeley DB databases
db_load	Is used to create database files from plain-text files
db_printlog	Converts database log files to human readable text
db_recover	Is used to restore a database to a consistent state after a failure
db_stat	Displays statistics for Berkeley databases
db_upgrade	Is used to upgrade database files to a newer version of Berkeley DB
db_verify	Is used to run consistency checks on database files
`libdb.{so,a}`	Contains functions to manipulate database files from C programs
`libdb_cxx.{so,a}`	Contains functions to manipulate database files from C++ programs

6.16. Sed-4.1.5

The Sed package contains a stream editor.

Approximate build time: 0.2 SBU
Required disk space: 10 MB

6.16.1. Installation of Sed

Prepare Sed for compilation:

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

The meaning of the new configure option:

--enable-html
This builds the HTML documentation.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.16.2. Contents of Sed

Installed program: sed

Short Descriptions

sed Filters and transforms text files in a single pass

6.17. E2fsprogs-1.41.3

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

Approximate build time: 0.7 SBU testsuite included
Required disk space: 54 MB testsuite included

6.17.1. Installation of E2fsprogs

Fix a hardcoded path to `/bin/rm` in E2fsprogs' testsuite:

```
sed -i 's@/bin/rm@/tools&@' lib/blkid/test_probe.in
```

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

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

Prepare E2fsprogs for compilation:

```
../configure --prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs
```

The meaning of the configure options:

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

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

One of the E2fsprogs tests will attempt to allocate 256 MB of memory. If you do not have significantly more RAM than this, it is recommended to enable sufficient swap space for the test. See Section 2.3, “Creating a File System on the Partition” and Section 2.4, “Mounting the New Partition” for details on creating and enabling swap space.

Install the binaries, documentation, and shared libraries:

```
make install
```

Install the static libraries and headers:

```
make install-libs
```

Make the installed static libraries writable so debugging symbols can be removed later.

```
chmod -v u+w /usr/lib/{libblkid,libcom_err,libe2p,libext2fs,libss,libuuid}.a
```

This package installs a gzipped .info file but doesn't update the system-wide dir file. Unzip this file and then update the system dir file using the following commands.

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir \
             /usr/share/info/libext2fs.info
```

If desired, create and install some additional documentation by issuing the following commands:

```
makeinfo -o      doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir \
             /usr/share/info/com_err.info

install -v -m644 -D ../doc/libblkid.txt \
        /usr/share/doc/e2fsprogs-1.41.3/libblkid.txt
```

6.17.2. Contents of E2fsprogs

Installed programs: badblocks, blkid, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, e2undo, filefrag, findfs, fsck, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, fsck.ext4, fsck.ext4dev, mklost+found, resize2fs, tune2fs, uuidd, and uuidgen.

Installed libraries: libblkid.{a,so}, libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libss.{a,so}, and libuuid.{a,so}

Short Descriptions

badblocks Searches a device (usually a disk partition) for bad blocks

blkid A command line utility to locate and print block device attributes

chattr Changes the attributes of files on an ext2 file system; it also changes ext3 file systems, the journaling version of ext2 file systems

compile_et An error table compiler; it converts a table of error-code names and messages into a C source file suitable for use with the com_err library

debugfs A file system debugger; it can be used to examine and change the state of an ext2 file system

dumpe2fs Prints the super block and blocks group information for the file system present on a given device

e2fsck Is used to check, and optionally repair ext2 file systems and ext3 file systems

e2image Is used to save critical ext2 file system data to a file

e2label Displays or changes the file system label on the ext2 file system present on a given device

e2undo Replays the undo log undo_log for an ext2/ext3/ext4 filesystem found on a device. This can be used to undo a failed operation by an e2fsprogs program.

filefrag Reports on how badly fragmented a particular file might be

findfs Finds a file system by label or Universally Unique Identifier (UUID)

fsck Is used to check, and optionally repair, file systems

fsck.ext2 By default checks ext2 file systems. This is a hard link to **fsck**.

fsck.ext3	By default checks `ext3` file systems. This is a hard link to **fsck**.
fsck.ext4	By default checks `ext4` file systems. This is a hard link to **fsck**.
fsck.ext4dev	By default checks `ext4` development file systems. This is a hard link to **fsck**.
logsave	Saves the output of a command in a log file
lsattr	Lists the attributes of files on a second extended file system
mk_cmds	Converts a table of command names and help messages into a C source file suitable for use with the `libss` subsystem library
mke2fs	Creates an `ext2` or `ext3` file system on the given device
mkfs.ext2	By default creates `ext2` file systems. This is a hard link to **mke2fs**.
mkfs.ext3	By default creates `ext3` file systems. This is a hard link to **mke2fs**.
mkfs.ext4	By default creates `ext4` file systems. This is a hard link to **mke2fs**.
mkfs.ext4dev	By default creates `ext4` development file systems. This is a hard link to **mke2fs**.
mklost+found	Used to create a `lost+found` directory on an `ext2` file system; it pre-allocates disk blocks to this directory to lighten the task of **e2fsck**
resize2fs	Can be used to enlarge or shrink an `ext2` file system
tune2fs	Adjusts tunable file system parameters on an `ext2` file system
uuidd	A daemon used by the UUID library to generate time-based UUIDs in a secure and guranteed-unique fashion.
uuidgen	Creates new UUIDs. Each new UUID can reasonably be considered unique among all UUIDs created, on the local system and on other systems, in the past and in the future
`libblkid`	Contains routines for device identification and token extraction
`libcom_err`	The common error display routine
`libe2p`	Used by **dumpe2fs**, **chattr**, and **lsattr**
`libext2fs`	Contains routines to enable user-level programs to manipulate an `ext2` file system
`libss`	Used by **debugfs**
`libuuid`	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

6.18. Coreutils-6.12

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

Approximate build time: 1.7 SBU testsuite included
Required disk space: 89 MB testsuite included

6.18.1. Installation of Coreutils

A known issue with the **uname** program from this package is that the *-p* switch always returns `unknown`. The following patch fixes this behavior for Intel architectures:

```
patch -Np1 -i ../coreutils-6.12-uname-1.patch
```

There's an internal issue with Coreutils which makes some of the programs behave abnormally if you build using an older kernel. Apply a patch to fix the issue:

```
patch -Np1 -i ../coreutils-6.12-old_build_kernel-1.patch
```

POSIX requires that programs from Coreutils recognize character boundaries correctly even in multibyte locales. The following patch fixes this non-compliance and other internationalization-related bugs:

```
patch -Np1 -i ../coreutils-6.12-i18n-2.patch
```

Note

In the past, many bugs were found in this patch. When reporting new bugs to Coreutils maintainers, please check first if they are reproducible without this patch.

Now prepare Coreutils for compilation:

```
./configure --prefix=/usr --enable-install-program=hostname \
    --enable-no-install-program=kill,uptime
```

The meaning of the configure options:

--enable-no-install-program=kill,uptime
The purpose of this switch is to prevent Coreutils from installing binaries that will be installed by other packages later.

Compile the package:

```
make
```

Skip down to “Install the package” if not running the test suite.

Now the test suite is ready to be run. First, run the tests that are meant to be run as user `root`:

```
make NON_ROOT_USERNAME=nobody check-root
```

We're going to run the remainder of the tests as the `nobody` user. Certain tests, however, require that the user be a member of more than one group. So that these tests are not skipped we'll add a temporary group and make the user `nobody` a part of it:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Fix some of the permissions so that the non-root user can compile and run the tests:

```
chown -Rv nobody config.log {gnulib-tests,lib,src}/.deps
```

Now run the tests:

```
su-tools nobody -s /bin/bash -c "make RUN_EXPENSIVE_TESTS=yes check"
```

Remove the temporary group:

```
sed -i '/dummy/d' /etc/group
```

Install the package:

```
make install
```

Move programs to the locations specified by the FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,hostname,ln,ls,mkdir,mknod,mv,pwd,readlink,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

Some of the scripts in the LFS-Bootscripts package depend on **head**, **sleep**, and **nice**. As `/usr` may not be available during the early stages of booting, those binaries need to be on the root partition:

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.18.2. Contents of Coreutils

Installed programs: base64, basename, cat, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stty, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, and yes

Short Descriptions

base64	Encodes and decodes data according to the base64 (RFC 3548) specification
basename	Strips any path and a given suffix from a file name
cat	Concatenates files to standard output
chgrp	Changes the group ownership of files and directories
chmod	Changes the permissions of each file to the given mode; the mode can be either a symbolic representation of the changes to make or an octal number representing the new permissions
chown	Changes the user and/or group ownership of files and directories
chroot	Runs a command with the specified directory as the `/` directory
cksum	Prints the Cyclic Redundancy Check (CRC) checksum and the byte counts of each specified file

comm	Compares two sorted files, outputting in three columns the lines that are unique and the lines that are common
cp	Copies files
csplit	Splits a given file into several new files, separating them according to given patterns or line numbers and outputting the byte count of each new file
cut	Prints sections of lines, selecting the parts according to given fields or positions
date	Displays the current time in the given format, or sets the system date
dd	Copies a file using the given block size and count, while optionally performing conversions on it
df	Reports the amount of disk space available (and used) on all mounted file systems, or only on the file systems holding the selected files
dir	Lists the contents of each given directory (the same as the **ls** command)
dircolors	Outputs commands to set the `LS_COLOR` environment variable to change the color scheme used by **ls**
dirname	Strips the non-directory suffix from a file name
du	Reports the amount of disk space used by the current directory, by each of the given directories (including all subdirectories) or by each of the given files
echo	Displays the given strings
env	Runs a command in a modified environment
expand	Converts tabs to spaces
expr	Evaluates expressions
factor	Prints the prime factors of all specified integer numbers
false	Does nothing, unsuccessfully; it always exits with a status code indicating failure
fmt	Reformats the paragraphs in the given files
fold	Wraps the lines in the given files
groups	Reports a user's group memberships
head	Prints the first ten lines (or the given number of lines) of each given file
hostid	Reports the numeric identifier (in hexadecimal) of the host
hostname	Reports or sets the name of the host
id	Reports the effective user ID, group ID, and group memberships of the current user or specified user
install	Copies files while setting their permission modes and, if possible, their owner and group
join	Joins the lines that have identical join fields from two separate files
link	Creates a hard link with the given name to a file
ln	Makes hard links or soft (symbolic) links between files
logname	Reports the current user's login name
ls	Lists the contents of each given directory
md5sum	Reports or checks Message Digest 5 (MD5) checksums
mkdir	Creates directories with the given names
mkfifo	Creates First-In, First-Outs (FIFOs), a “named pipe” in UNIX parlance, with the given names

mknod	Creates device nodes with the given names; a device node is a character special file, a block special file, or a FIFO
mktemp	Creates temporary files in a secure manner; it is used in scripts
mv	Moves or renames files or directories
nice	Runs a program with modified scheduling priority
nl	Numbers the lines from the given files
nohup	Runs a command immune to hangups, with its output redirected to a log file
od	Dumps files in octal and other formats
paste	Merges the given files, joining sequentially corresponding lines side by side, separated by tab characters
pathchk	Checks if file names are valid or portable
pinky	Is a lightweight finger client; it reports some information about the given users
pr	Paginates and columnates files for printing
printenv	Prints the environment
printf	Prints the given arguments according to the given format, much like the C printf function
ptx	Produces a permuted index from the contents of the given files, with each keyword in its context
pwd	Reports the name of the current working directory
readlink	Reports the value of the given symbolic link
rm	Removes files or directories
rmdir	Removes directories if they are empty
seq	Prints a sequence of numbers within a given range and with a given increment
sha1sum	Prints or checks 160-bit Secure Hash Algorithm 1 (SHA1) checksums
sha224sum	Prints or checks 224-bit Secure Hash Algorithm checksums
sha256sum	Prints or checks 256-bit Secure Hash Algorithm checksums
sha384sum	Prints or checks 384-bit Secure Hash Algorithm checksums
sha512sum	Prints or checks 512-bit Secure Hash Algorithm checksums
shred	Overwrites the given files repeatedly with complex patterns, making it difficult to recover the data
shuf	Shuffles lines of text
sleep	Pauses for the given amount of time
sort	Sorts the lines from the given files
split	Splits the given file into pieces, by size or by number of lines
stat	Displays file or filesystem status
stty	Sets or reports terminal line settings
sum	Prints checksum and block counts for each given file
sync	Flushes file system buffers; it forces changed blocks to disk and updates the super block
tac	Concatenates the given files in reverse

tail	Prints the last ten lines (or the given number of lines) of each given file
tee	Reads from standard input while writing both to standard output and to the given files
test	Compares values and checks file types
touch	Changes file timestamps, setting the access and modification times of the given files to the current time; files that do not exist are created with zero length
tr	Translates, squeezes, and deletes the given characters from standard input
true	Does nothing, successfully; it always exits with a status code indicating success
tsort	Performs a topological sort; it writes a completely ordered list according to the partial ordering in a given file
tty	Reports the file name of the terminal connected to standard input
uname	Reports system information
unexpand	Converts spaces to tabs
uniq	Discards all but one of successive identical lines
unlink	Removes the given file
users	Reports the names of the users currently logged on
vdir	Is the same as **ls -l**
wc	Reports the number of lines, words, and bytes for each given file, as well as a total line when more than one file is given
who	Reports who is logged on
whoami	Reports the user name associated with the current effective user ID
yes	Repeatedly outputs “y” or a given string until killed

6.19. Iana-Etc-2.30

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

Approximate build time:	less than 0.1 SBU
Required disk space:	2.1 MB

6.19.1. Installation of Iana-Etc

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

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.19.2. Contents of Iana-Etc

Installed files:	/etc/protocols and /etc/services

Short Descriptions

`/etc/protocols`	Describes the various DARPA Internet protocols that are available from the TCP/IP subsystem
`/etc/services`	Provides a mapping between friendly textual names for internet services, and their underlying assigned port numbers and protocol types

6.20. M4-1.4.12

The M4 package contains a macro processor.

Approximate build time: 0.3 SBU testsuite included
Required disk space: 12 MB

6.20.1. Installation of M4

Prepare M4 for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.20.2. Contents of M4

Installed program: m4

Short Descriptions

m4 copies the given files while expanding the macros that they contain. These macros are either built-in or user-defined and can take any number of arguments. Besides performing macro expansion, **m4** has built-in functions for including named files, running Unix commands, performing integer arithmetic, manipulating text, recursion, etc. The **m4** program can be used either as a front-end to a compiler or as a macro processor in its own right.

6.21. Bison-2.3

The Bison package contains a parser generator.

Approximate build time: 0.2 SBU
Required disk space: 12.3 MB

6.21.1. Installation of Bison

Prepare Bison for compilation:

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

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

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

Compile the package:

```
make
```

To test the results (about 0.5 SBU), issue:

```
make check
```

Install the package:

```
make install
```

6.21.2. Contents of Bison

Installed programs: bison and yacc
Installed library: liby.a

Short Descriptions

bison Generates, from a series of rules, a program for analyzing the structure of text files; Bison is a replacement for Yacc (Yet Another Compiler Compiler)

yacc A wrapper for **bison**, meant for programs that still call **yacc** instead of **bison**; it calls **bison** with the `-y` option

`liby.a` The Yacc library containing implementations of Yacc-compatible `yyerror` and `main` functions; this library is normally not very useful, but POSIX requires it

6.22. Ncurses-5.6

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

Approximate build time: 0.7 SBU
Required disk space: 31 MB

6.22.1. Installation of Ncurses

Apply the following patch to fix a number of issues uncovered by the static code analysis tool, Coverity:

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

Prepare Ncurses for compilation:

```
./configure --prefix=/usr --with-shared --without-debug --enable-widec
```

The meaning of the configure option:

--enable-widec
This switch causes wide-character libraries (e.g., `libncursesw.so.5.6`) to be built instead of normal ones (e.g., `libncurses.so.5.6`). These wide-character libraries are usable in both multibyte and traditional 8-bit locales, while normal libraries work properly only in 8-bit locales. Wide-character and normal libraries are source-compatible, but not binary-compatible.

Compile the package:

```
make
```

This package has a test suite, but it can only be run after the package has been installed. The tests reside in the `test/` directory. See the `README` file in that directory for further details.

Install the package:

```
make install
```

Correct the permissions of a library that should not be executable:

```
chmod -v 644 /usr/lib/libncurses++w.a
```

Move the libraries to the `/lib` directory, where they are expected to reside:

```
mv -v /usr/lib/libncursesw.so.5* /lib
```

Because the libraries have been moved, one symlink points to a non-existent file. Recreate it:

```
ln -sfv ../../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

Many applications still expect the linker to be able to find non-wide-character Ncurses libraries. Trick such applications into linking with wide-character libraries by means of symlinks and linker scripts:

```
for lib in curses ncurses form panel menu ; do \
    rm -vf /usr/lib/lib${lib}.so ; \
    echo "INPUT(-l${lib}w)" >/usr/lib/lib${lib}.so ; \
    ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a ; \
done
ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

Finally, make sure that old applications that look for `-lcurses` at build time are still buildable:

```
rm -vf /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" >/usr/lib/libcursesw.so
ln -sfv libncurses.so /usr/lib/libcurses.so
ln -sfv libncursesw.a /usr/lib/libcursesw.a
ln -sfv libncurses.a /usr/lib/libcurses.a
```

If desired, install the Ncurses documentation:

```
mkdir -v       /usr/share/doc/ncurses-5.6
cp -v -R doc/* /usr/share/doc/ncurses-5.6
```

Note

The instructions above don't create non-wide-character Ncurses libraries since no package installed by compiling from sources would link against them at runtime. If you must have such libraries because of some binary-only application, build them with the following commands:

```
make distclean
./configure --prefix=/usr --with-shared --without-normal \
  --without-debug --without-cxx-binding
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.22.2. Contents of Ncurses

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

Installed libraries: libcursesw.{a,so} (symlink and linker script to libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} and their non-wide-character counterparts without "w" in the library names.

Short Descriptions

captoinfo	Converts a termcap description into a terminfo description
clear	Clears the screen, if possible
infocmp	Compares or prints out terminfo descriptions
infotocap	Converts a terminfo description into a termcap description
ncurses5-config	Provides configuration information for ncurses
reset	Reinitializes a terminal to its default values
tack	The terminfo action checker; it is mainly used to test the accuracy of an entry in the terminfo database
tic	The terminfo entry-description compiler that translates a terminfo file from source format into the binary format needed for the ncurses library routines. A terminfo file contains information on the capabilities of a certain terminal
toe	Lists all available terminal types, giving the primary name and description for each

tput	Makes the values of terminal-dependent capabilities available to the shell; it can also be used to reset or initialize a terminal or report its long name
tset	Can be used to initialize terminals
`libcurses`	A link to `libncurses`
`libncurses`	Contains functions to display text in many complex ways on a terminal screen; a good example of the use of these functions is the menu displayed during the kernel's **make menuconfig**
`libform`	Contains functions to implement forms
`libmenu`	Contains functions to implement menus
`libpanel`	Contains functions to implement panels

6.23. Procps-3.2.7

The Procps package contains programs for monitoring processes.

Approximate build time: 0.1 SBU
Required disk space: 2.3 MB

6.23.1. Installation of Procps

Apply a patch to fix a unicode related issue in the **watch** program:

```
patch -Np1 -i ../procps-3.2.7-watch_unicode-1.patch
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.23.2. Contents of Procps

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

Short Descriptions

free	Reports the amount of free and used memory (both physical and swap memory) in the system
kill	Sends signals to processes
pgrep	Looks up processes based on their name and other attributes
pkill	Signals processes based on their name and other attributes
pmap	Reports the memory map of the given process
ps	Lists the current running processes
pwdx	Reports the current working directory of a process
skill	Sends signals to processes matching the given criteria
slabtop	Displays detailed kernel slap cache information in real time
snice	Changes the scheduling priority of processes matching the given criteria
sysctl	Modifies kernel parameters at run time
tload	Prints a graph of the current system load average
top	Displays a list of the most CPU intensive processes; it provides an ongoing look at processor activity in real time
uptime	Reports how long the system has been running, how many users are logged on, and the system load averages

vmstat Reports virtual memory statistics, giving information about processes, memory, paging, block Input/Output (IO), traps, and CPU activity

w Shows which users are currently logged on, where, and since when

watch Runs a given command repeatedly, displaying the first screen-full of its output; this allows a user to watch the output change over time

`libproc` Contains the functions used by most programs in this package

6.24. Libtool-2.2.6a

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

Approximate build time: 0.1 SBU
Required disk space: 36 MB testsuite included

6.24.1. Installation of Libtool

Prepare Libtool for compilation:

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

Compile the package:

```
make
```

To test the results (about 3.0 SBU), issue:

```
make check
```

Install the package:

```
make install
```

6.24.2. Contents of Libtool

Installed programs: libtool and libtoolize
Installed libraries: libltdl.{a,so}

Short Descriptions

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

6.25. Zlib-1.2.3

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

Approximate build time: less than 0.1 SBU
Required disk space: 3.1 MB

6.25.1. Installation of Zlib

Note

Zlib is known to build its shared library incorrectly if `CFLAGS` is specified in the environment. If using a specified `CFLAGS` variable, be sure to add the *`-fPIC`* directive to the `CFLAGS` variable for the duration of the configure command below, then remove it afterwards.

Prepare Zlib for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the shared library:

```
make install
```

The previous command installed a `.so` file in `/lib`. We will remove it and relink it into `/usr/lib`:

```
rm -v /lib/libz.so
ln -sfv ../../lib/libz.so.1.2.3 /usr/lib/libz.so
```

Build the static library:

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

To test the results again, issue:

```
make check
```

Install the static library:

```
make install
```

Fix the permissions on the static library:

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

6.25.2. Contents of Zlib

Installed libraries:	libz.{a,so}

Short Descriptions

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

6.26. Perl-5.10.0

The Perl package contains the Practical Extraction and Report Language.

Approximate build time: 2.5 SBU
Required disk space: 178 MB testsuite included

6.26.1. Installation of Perl

First create a basic `/etc/hosts` file to be referenced in one of Perl's configuration files as well as the optional testsuite:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

The following patch fixes known vulnerabilities and other issues identified by the developers:

```
patch -Np1 -i ../perl-5.10.0-consolidated-1.patch
```

This version of Perl now builds the Compress::Raw::Zlib module. By default Perl will use an internal copy of the Zlib source for the build. Issue the following command so that Perl will use the Zlib library installed on the system:

```
sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|"           \
       -e "s|INCLUDE\s*= ./zlib-src|INCLUDE    = /usr/include|" \
       -e "s|LIB\s*= ./zlib-src|LIB        = /usr/lib|"         \
    ext/Compress/Raw/Zlib/config.in
```

To have full control over the way Perl is set up, you can run the interactive **Configure** script and hand-pick the way this package is built. If you prefer, you can use the defaults that Perl auto-detects, by preparing Perl for compilation with:

```
sh Configure -des -Dprefix=/usr \
                  -Dvendorprefix=/usr           \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"
```

The meaning of the configure options:

-Dvendorprefix=/usr
: This ensures **perl** knows how to tell packages where they should install their perl modules.

-Dpager="/usr/bin/less -isR"
: This corrects an error in the way that **perldoc** invokes the **less** program.

-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3
: Since Groff is not installed yet, **Configure** thinks that we do not want man pages for Perl. Issuing these parameters overrides this decision.

Compile the package:

```
make
```

To test the results (approximately 2.5 SBU), issue:

```
make test
```

Install the package:

```
make install
```

6.26.2. Contents of Perl

Installed programs:	a2p, c2ph, cpan, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, libnetcfg, perl, perl5.10.0 (link to perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), s2p, splain, and xsubpp
Installed libraries:	Several hundred which cannot all be listed here

Short Descriptions

a2p	Translates awk to Perl
c2ph	Dumps C structures as generated from **cc -g -S**
cpan	Interact with the Comprehensive Perl Archive Network (CPAN) from the command line
dprofpp	Displays Perl profile data
enc2xs	Builds a Perl extension for the Encode module from either Unicode Character Mappings or Tcl Encoding Files
find2perl	Translates **find** commands to Perl
h2ph	Converts `.h` C header files to `.ph` Perl header files
h2xs	Converts `.h` C header files to Perl extensions
instmodsh	Shell script for examining installed Perl modules, and can even create a tarball from an installed module
libnetcfg	Can be used to configure the `libnet`
perl	Combines some of the best features of C, **sed**, **awk** and **sh** into a single swiss-army language
perl5.10.0	A hard link to **perl**
perlbug	Used to generate bug reports about Perl, or the modules that come with it, and mail them
perlcc	Generates executables from Perl programs
perldoc	Displays a piece of documentation in pod format that is embedded in the Perl installation tree or in a Perl script
perlivp	The Perl Installation Verification Procedure; it can be used to verify that Perl and its libraries have been installed correctly
piconv	A Perl version of the character encoding converter **iconv**
pl2pm	A rough tool for converting Perl4 `.pl` files to Perl5 `.pm` modules
pod2html	Converts files from pod format to HTML format
pod2latex	Converts files from pod format to LaTeX format
pod2man	Converts pod data to formatted *roff input
pod2text	Converts pod data to formatted ASCII text
pod2usage	Prints usage messages from embedded pod docs in files

podchecker	Checks the syntax of pod format documentation files
podselect	Displays selected sections of pod documentation
prove	Command line tool for running tests against the Test::Harness module.
psed	A Perl version of the stream editor **sed**
pstruct	Dumps C structures as generated from **cc -g -S** stabs
s2p	Translates **sed** scripts to Perl
splain	Is used to force verbose warning diagnostics in Perl
xsubpp	Converts Perl XS code into C code

6.27. Readline-5.2

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

Approximate build time: 0.1 SBU
Required disk space: 10.2 MB

6.27.1. Installation of Readline

Reinstalling Readline will cause the old libraries to be moved to <libraryname>.old. While this is normally not a problem, in some cases it can trigger a linking bug in **ldconfig**. This can be avoided by issuing the following two seds:

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

Readline contains a bug in its handling of non-multibyte characters, which can lead to incorrect display calculations and incorrect redisplay. Fix this issue by applying the following patch from the upstream maintainer:

```
patch -Np1 -i ../readline-5.2-fixes-5.patch
```

Prepare Readline for compilation:

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

Compile the package:

```
make SHLIB_LIBS=-lncurses
```

The meaning of the make option:

SHLIB_LIBS=-lncurses
This option forces Readline to link against the `libncurses` (really, `libncursesw`) library.

This package does not come with a test suite.

Install the package:

```
make install
```

Now move the static libraries to a more appropriate location:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Next, remove the `.so` files in `/lib` and relink them into `/usr/lib`:

```
rm -v /lib/lib{readline,history}.so
ln -sfv ../../lib/libreadline.so.5 /usr/lib/libreadline.so
ln -sfv ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

If desired, install the documentation:

```
mkdir   -v       /usr/share/doc/readline-5.2
install -v -m644 doc/*.{ps,pdf,html,dvi} \
                 /usr/share/doc/readline-5.2
```

6.27.2. Contents of Readline

Installed libraries: libhistory.{a,so}, and libreadline.{a,so}

Short Descriptions

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

6.28. Autoconf-2.63

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

Approximate build time: less than 0.1 SBU
Required disk space: 14.3 MB testsuite included

6.28.1. Installation of Autoconf

Prepare Autoconf for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

This takes a long time, about 4.7 SBUs. In addition, 6 tests are skipped that use Automake. For full test coverage, Autoconf can be re-tested after Automake has been installed.

Install the package:

```
make install
```

6.28.2. Contents of Autoconf

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

Short Descriptions

autoconf Produces shell scripts that automatically configure software source code packages to adapt to many kinds of Unix-like systems. The configuration scripts it produces are independent—running them does not require the **autoconf** program.

autoheader A tool for creating template files of C *#define* statements for configure to use

autom4te A wrapper for the M4 macro processor

autoreconf Automatically runs **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, and **libtoolize** in the correct order to save time when changes are made to **autoconf** and **automake** template files

autoscan Helps to create a `configure.in` file for a software package; it examines the source files in a directory tree, searching them for common portability issues, and creates a `configure.scan` file that serves as as a preliminary `configure.in` file for the package

autoupdate Modifies a `configure.in` file that still calls **autoconf** macros by their old names to use the current macro names

ifnames Helps when writing `configure.in` files for a software package; it prints the identifiers that the package uses in C preprocessor conditionals. If a package has already been set up to have some portability, this program can help determine what **configure** needs to check for. It can also fill in gaps in a `configure.in` file generated by **autoscan**

6.29. Automake-1.10.1

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

Approximate build time: less than 0.1 SBU
Required disk space: 7.9 MB

6.29.1. Installation of Automake

Patch a test in the Automake test suite to fix an issue that occurs when running the tests as `root`:

```
patch -Np1 -i ../automake-1.10.1-test_fix-1.patch
```

Prepare Automake for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.10.1
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

This takes a long time, about 10 SBUs.

Install the package:

```
make install
```

6.29.2. Contents of Automake

Installed programs: acinstall, aclocal, aclocal-1.10.1, automake, automake-1.10.1, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree, and ylwrap

Short Descriptions

acinstall A script that installs aclocal-style M4 files

aclocal Generates `aclocal.m4` files based on the contents of `configure.in` files

aclocal-1.10.1 A hard link to **aclocal**

automake A tool for automatically generating `Makefile.in` files from `Makefile.am` files. To create all the `Makefile.in` files for a package, run this program in the top-level directory. By scanning the `configure.in` file, it automatically finds each appropriate `Makefile.am` file and generates the corresponding `Makefile.in` file

automake-1.10.1 A hard link to **automake**

compile A wrapper for compilers

config.guess A script that attempts to guess the canonical triplet for the given build, host, or target architecture

config.sub A configuration validation subroutine script

depcomp	A script for compiling a program so that dependency information is generated in addition to the desired output
elisp-comp	Byte-compiles Emacs Lisp code
install-sh	A script that installs a program, script, or data file
mdate-sh	A script that prints the modification time of a file or directory
missing	A script acting as a common stub for missing GNU programs during an installation
mkinstalldirs	A script that creates a directory tree
py-compile	Compiles a Python program
symlink-tree	A script to create a symlink tree of a directory tree
ylwrap	A wrapper for **lex** and **yacc**

6.30. Bash-3.2

The Bash package contains the Bourne-Again SHell.

Approximate build time: 0.4 SBU
Required disk space: 25.8 MB

6.30.1. Installation of Bash

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

```
tar -xvf ../bash-doc-3.2.tar.gz
sed -i "s|htmldir = @htmldir@|htmldir = /usr/share/doc/bash-3.2|" \
    Makefile.in
```

Apply fixes for several bugs discovered since the initial release of Bash-3.2:

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

Prepare Bash for compilation:

```
./configure --prefix=/usr --bindir=/bin \
    --without-bash-malloc \
    --with-installed-readline ac_cv_func_working_mktime=yes
```

The meaning of the configure options:

--with-installed-readline
: This option tells Bash to use the `readline` library that is already installed on the system rather than using its own readline version.

Compile the package:

```
make
```

Skip down to “Install the package” if not running the test suite.

To prepare the tests, ensure that the locale setting from our environment will be used and that the `nobody` user can read the standard input device and write to the sources tree:

```
sed -i 's/LANG/LC_ALL/' tests/intl.tests
sed -i 's@tests@& </dev/tty@' tests/run-test
chown -Rv nobody ./
```

Now, run the tests as the `nobody` user:

```
su-tools nobody -s /bin/bash -c "make tests"
```

Install the package:

```
make install
```

Run the newly compiled **bash** program (replacing the one that is currently being executed):

```
exec /bin/bash --login +h
```

Note

The parameters used make the **bash** process an interactive login shell and continue to disable hashing so that new programs are found as they become available.

6.30.2. Contents of Bash

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

Short Descriptions

bash	A widely-used command interpreter; it performs many types of expansions and substitutions on a given command line before executing it, thus making this interpreter a powerful tool
bashbug	A shell script to help the user compose and mail standard formatted bug reports concerning **bash**
sh	A symlink to the **bash** program; when invoked as **sh**, **bash** tries to mimic the startup behavior of historical versions of **sh** as closely as possible, while conforming to the POSIX standard as well

6.31. Bzip2-1.0.5

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

Approximate build time: less than 0.1 SBU
Required disk space: 6.5 MB

6.31.1. Installation of Bzip2

Apply a patch to install the documentation for this package:

```
patch -Np1 -i ../bzip2-1.0.5-install_docs-1.patch
```

Prepare Bzip2 for compilation with:

```
make -f Makefile-libbz2_so
make clean
```

The meaning of the make parameter:

`-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.

Compile and test the package:

```
make
```

Install the programs:

```
make PREFIX=/usr install
```

Install the shared **bzip2** binary into the `/bin` directory, make some necessary symbolic links, and clean up:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.31.2. Contents of Bzip2

Installed programs: bunzip2 (link to bzip2), bzcat (link to bzip2), bzcmp (link to bzdiff), bzdiff, bzegrep (link to bzgrep), bzfgrep (link to bzgrep), bzgrep, bzip2, bzip2recover, bzless (link to bzmore), and bzmore
Installed libraries: libbz2.{a,so}

Short Descriptions

bunzip2	Decompresses bzipped files
bzcat	Decompresses to standard output

bzcmp	Runs **cmp** on bzipped files
bzdiff	Runs **diff** on bzipped files
bzgrep	Runs **grep** on bzipped files
bzegrep	Runs **egrep** on bzipped files
bzfgrep	Runs **fgrep** on bzipped files
bzip2	Compresses files using the Burrows-Wheeler block sorting text compression algorithm with Huffman coding; the compression rate is better than that achieved by more conventional compressors using “Lempel-Ziv” algorithms, like **gzip**
bzip2recover	Tries to recover data from damaged bzipped files
bzless	Runs **less** on bzipped files
bzmore	Runs **more** on bzipped files
`libbz2*`	The library implementing lossless, block-sorting data compression, using the Burrows-Wheeler algorithm

6.32. Diffutils-2.8.1

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

Approximate build time: 0.1 SBU
Required disk space: 6.3 MB

6.32.1. Installation of Diffutils

POSIX requires the **diff** command to treat whitespace characters according to the current locale. The following patch fixes the non-compliance issue:

```
patch -Np1 -i ../diffutils-2.8.1-i18n-1.patch
```

The above patch will cause the Diffutils build system to attempt to rebuild the `diff.1` man page using the unavailable program **help2man**. The result is an unreadable man page for **diff**. We can avoid this by updating the timestamp on the file `man/diff.1`:

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

Prepare Diffutils for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.32.2. Contents of Diffutils

Installed programs: cmp, diff, diff3, and sdiff

Short Descriptions

cmp Compares two files and reports whether or in which bytes they differ

diff Compares two files or directories and reports which lines in the files differ

diff3 Compares three files line by line

sdiff Merges two files and interactively outputs the results

6.33. File-4.26

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

Approximate build time: 0.1 SBU
Required disk space: 8.9 MB

6.33.1. Installation of File

Fix the man page so that it will reflect recent changes to the *-e* (*--exclude*) parameter:

```
sed -i -e '197,+1d' \
       -e '189,+1d' \
       -e 's/token$/tokens/' doc/file.man
```

Prepare File for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.33.2. Contents of File

Installed programs: file
Installed library: libmagic.{a,so}

Short Descriptions

file Tries to classify each given file; it does this by performing several tests—file system tests, magic number tests, and language tests

`libmagic` Contains routines for magic number recognition, used by the **file** program

6.34. Gawk-3.1.6

The Gawk package contains programs for manipulating text files.

Approximate build time:	0.3 SBU
Required disk space:	21 MB

6.34.1. Installation of Gawk

Prepare Gawk for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib \
   ac_cv_func_working_mktime=yes
```

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

If desired, install the documentation:

```
mkdir -v /usr/share/doc/gawk-3.1.6
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} \
         /usr/share/doc/gawk-3.1.6
```

6.34.2. Contents of Gawk

Installed programs:	awk (link to gawk), gawk, gawk-3.1.6, grcat, igawk, pgawk, pgawk-3.1.6, and pwcat

Short Descriptions

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

6.35. Findutils-4.4.0

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

Approximate build time: 0.4 SBU
Required disk space: 22 MB

6.35.1. Installation of Findutils

Prepare Findutils for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib/findutils \
    --localstatedir=/var/lib/locate
```

The meaning of the configure options:

--localstatedir
: This option changes the location of the **locate** database to be in `/var/lib/locate`, which is FHS-compliant.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Some of the scripts in the LFS-Bootscripts package depend on **find**. As `/usr` may not be available during the early stages of booting, this program needs to be on the root partition. The **updatedb** script also needs to be modified to correct an explicit path:

```
mv -v /usr/bin/find /bin
sed -i -e 's/find:=${BINDIR}/find:=\/bin/' /usr/bin/updatedb
```

6.35.2. Contents of Findutils

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

Short Descriptions

bigram	Was formerly used to produce **locate** databases
code	Was formerly used to produce **locate** databases; it is the ancestor of **frcode**.
find	Searches given directory trees for files matching the specified criteria
frcode	Is called by **updatedb** to compress the list of file names; it uses front-compression, reducing the database size by a factor of four to five.
locate	Searches through a database of file names and reports the names that contain a given string or match a given pattern

updatedb Updates the **locate** database; it scans the entire file system (including other file systems that are currently mounted, unless told not to) and puts every file name it finds into the database

xargs Can be used to apply a given command to a list of files

6.36. Flex-2.5.35

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

Approximate build time: 0.2 SBU
Required disk space: 28 MB testsuite included

6.36.1. Installation of Flex

Prepare Flex for compilation:

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

Compile the package:

```
make
```

To test the results (about 0.5 SBU), issue:

```
make check
```

Install the package:

```
make install
```

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

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

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

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

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

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

If desired, install the `flex.pdf` documentation file:

```
mkdir -v /usr/share/doc/flex-2.5.35
cp    -v doc/flex.pdf \
         /usr/share/doc/flex-2.5.35
```

6.36.2. Contents of Flex

Installed programs: flex and lex
Installed library: libfl.a

Short Descriptions

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

6.37. GRUB-0.97

The GRUB package contains the GRand Unified Bootloader.

Approximate build time: 0.2 SBU
Required disk space: 10.2 MB

6.37.1. Installation of GRUB

This package is known to have issues when its default optimization flags (including the *-march* and *-mcpu* options) are changed. If any environment variables that override default optimizations have been defined, such as `CFLAGS` and `CXXFLAGS`, unset them when building GRUB.

Start by applying the following patch to allow for better drive detection, fix some GCC 4.x issues, and provide better SATA support for some disk controllers:

```
patch -Np1 -i ../grub-0.97-disk_geometry-1.patch
```

By default, GRUB doesn't support ext2 filesystems with 256-byte inodes. Fix this by applying the following patch:

```
patch -Np1 -i ../grub-0.97-256byte_inode-1.patch
```

Prepare GRUB for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
mkdir -v /boot/grub
cp -v /usr/lib/grub/i386-pc/stage{1,2} /boot/grub
```

Replace `i386-pc` with whatever directory is appropriate for the hardware in use.

The `i386-pc` directory contains a number of `*stage1_5` files, different ones for different file systems. Review the files available and copy the appropriate ones to the `/boot/grub` directory. Most users will copy the `e2fs_stage1_5` and/or `reiserfs_stage1_5` files.

6.37.2. Contents of GRUB

Installed programs: grub, grub-install, grub-md5-crypt, grub-set-default, grub-terminfo, and mbchk

Short Descriptions

grub	The Grand Unified Bootloader's command shell
grub-install	Installs GRUB on the given device

grub-md5-crypt	Encrypts a password in MD5 format
grub-set-default	Sets the default boot entry for GRUB
grub-terminfo	Generates a terminfo command from a terminfo name; it can be employed if an unknown terminal is being used
mbchk	Checks the format of a multi-boot kernel

6.38. Gettext-0.17

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

Approximate build time:	2.2 SBU
Required disk space:	128 MB

6.38.1. Installation of Gettext

Prepare Gettext for compilation:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/gettext-0.17
```

Compile the package:

```
make
```

To test the results (this takes a long time, around 3 SBUs), issue:

```
make check
```

Install the package:

```
make install
```

6.38.2. Contents of Gettext

Installed programs:	autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, and xgettext
Installed libraries:	libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so}, and libgettextsrc.so

Short Descriptions

autopoint	Copies standard Gettext infrastructure files into a source package
config.charset	Outputs a system-dependent table of character encoding aliases
config.rpath	Outputs a system-dependent set of variables, describing how to set the runtime search path of shared libraries in an executable
envsubst	Substitutes environment variables in shell format strings
gettext	Translates a natural language message into the user's language by looking up the translation in a message catalog
gettext.sh	Primarily serves as a shell function library for gettext
gettextize	Copies all standard Gettext files into the given top-level directory of a package to begin internationalizing it
hostname	Displays a network hostname in various forms
msgattrib	Filters the messages of a translation catalog according to their attributes and manipulates the attributes

msgcat	Concatenates and merges the given `.po` files
msgcmp	Compares two `.po` files to check that both contain the same set of msgid strings
msgcomm	Finds the messages that are common to to the given `.po` files
msgconv	Converts a translation catalog to a different character encoding
msgen	Creates an English translation catalog
msgexec	Applies a command to all translations of a translation catalog
msgfilter	Applies a filter to all translations of a translation catalog
msgfmt	Generates a binary message catalog from a translation catalog
msggrep	Extracts all messages of a translation catalog that match a given pattern or belong to some given source files
msginit	Creates a new `.po` file, initializing the meta information with values from the user's environment
msgmerge	Combines two raw translations into a single file
msgunfmt	Decompiles a binary message catalog into raw translation text
msguniq	Unifies duplicate translations in a translation catalog
ngettext	Displays native language translations of a textual message whose grammatical form depends on a number
recode-sr-latin	Recodes Serbian text from Cyrillic to Latin script
xgettext	Extracts the translatable message lines from the given source files to make the first translation template
`libasprintf`	defines the *autosprintf* class, which makes C formatted output routines usable in C++ programs, for use with the <*string*> strings and the <*iostream*> streams
`libgettextlib`	a private library containing common routines used by the various Gettext programs; these are not intended for general use
`libgettextpo`	Used to write specialized programs that process `.po` files; this library is used when the standard applications shipped with Gettext (such as **msgcomm**, **msgcmp**, **msgattrib**, and **msgen**) will not suffice
`libgettextsrc`	A private library containing common routines used by the various Gettext programs; these are not intended for general use

6.39. Grep-2.5.3

The Grep package contains programs for searching through files.

Approximate build time: 0.1 SBU
Required disk space: 7.2 MB

6.39.1. Installation of Grep

The current Grep package has many bugs, especially in the support of multibyte locales. The following consolidated patch from Debian fixes some of them, improves the number of individual tests which are passed, and much improves the speed in UTF-8 locales:

```
patch -Np1 -i ../grep-2.5.3-debian_fixes-1.patch
```

Upstream have fixed some of the documentation, and changed some of the tests and the expected results in the newest test scripts. This means that not so many of the individual tests fail:

```
patch -Np1 -i ../grep-2.5.3-upstream_fixes-1.patch
```

Prepare Grep for compilation:

```
./configure --prefix=/usr \
    --bindir=/bin \
    --without-included-regex
```

The meaning of the configure switch:

--without-included-regex
: The configure check for glibc's regex library is broken when building against glibc-2.8. This switch forces the use of glibc's regex library.

Compile the package:

```
make
```

To test the results, issue:

```
make check || true
```

There are known test failures in the **foad1.sh** and **fmbtest.sh** tests. The "|| true" construct is used to avoid build automated build scripts failing due to the test failures. A good run will show 2 failures from 14 tests, but if you look at the output you will see in excess of forty individual tests which failed - these are all in the new tests added since the previous version.

Install the package:

```
make install
```

6.39.2. Contents of Grep

Installed programs: egrep, fgrep, and grep

Short Descriptions

egrep Prints lines matching an extended regular expression

fgrep Prints lines matching a list of fixed strings

grep Prints lines matching a basic regular expression

6.40. Groff-1.18.1.4

The Groff package contains programs for processing and formatting text.

Approximate build time:	0.4 SBU
Required disk space:	39.2 MB

6.40.1. Installation of Groff

Apply the patch that adds the “ascii8” and “nippon” devices to Groff:

```
patch -Np1 -i ../groff-1.18.1.4-debian_fixes-1.patch
```

> **Note**
>
> These devices are used by Man-DB when formatting non-English manual pages that are not in the ISO-8859-1 encoding. Currently, there is no working patch for Groff-1.19.x that adds this functionality.

Many screen fonts don't have Unicode single quotes and dashes in them. Tell Groff to use the ASCII equivalents instead:

```
sed -i -e 's/2010/002D/' -e 's/2212/002D/' \
    -e 's/2018/0060/' -e 's/2019/0027/' font/devutf8/R.proto
```

Groff expects the environment variable `PAGE` to contain the default paper size. For users in the United States, *`PAGE=letter`* is appropriate. Elsewhere, *`PAGE=A4`* may be more suitable. While the default paper size is configured during compilation, it can be overridden later by echoing either “A4” or “letter” to the `/etc/papersize` file.

Prepare Groff for compilation:

```
PAGE=<paper_size> ./configure --prefix=/usr --enable-multibyte
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make docdir=/usr/share/doc/groff-1.18.1.4 install
```

Some documentation programs, such as **xman**, will not work properly without the following symlinks:

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.40.2. Contents of Groff

Installed programs:	addftinfo, afmtodit, eqn, eqn2graph, geqn (link to eqn), grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, refer, soelim, tbl, tfmtodit, and troff

Short Descriptions

addftinfo	Reads a troff font file and adds some additional font-metric information that is used by the **groff** system
afmtodit	Creates a font file for use with **groff** and **grops**
eqn	Compiles descriptions of equations embedded within troff input files into commands that are understood by **troff**
eqn2graph	Converts a troff EQN (equation) into a cropped image
geqn	A link to **eqn**
grn	A **groff** preprocessor for gremlin files
grodvi	A driver for **groff** that produces TeX dvi format
groff	A front-end to the groff document formatting system; normally, it runs the **troff** program and a post-processor appropriate for the selected device
groffer	Displays groff files and man pages on X and tty terminals
grog	Reads files and guesses which of the **groff** options `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, and `-t` are required for printing files, and reports the **groff** command including those options
grolbp	Is a **groff** driver for Canon CAPSL printers (LBP-4 and LBP-8 series laser printers)
grolj4	Is a driver for **groff** that produces output in PCL5 format suitable for an HP LaserJet 4 printer
grops	Translates the output of GNU **troff** to PostScript
grotty	Translates the output of GNU **troff** into a form suitable for typewriter-like devices
gtbl	A link to **tbl**
hpftodit	Creates a font file for use with **groff -Tlj4** from an HP-tagged font metric file
indxbib	Creates an inverted index for the bibliographic databases with a specified file for use with **refer**, **lookbib**, and **lkbib**
lkbib	Searches bibliographic databases for references that contain specified keys and reports any references found
lookbib	Prints a prompt on the standard error (unless the standard input is not a terminal), reads a line containing a set of keywords from the standard input, searches the bibliographic databases in a specified file for references containing those keywords, prints any references found on the standard output, and repeats this process until the end of input
mmroff	A simple preprocessor for **groff**
neqn	Formats equations for American Standard Code for Information Interchange (ASCII) output
nroff	A script that emulates the **nroff** command using **groff**
pfbtops	Translates a PostScript font in `.pfb` format to ASCII
pic	Compiles descriptions of pictures embedded within troff or TeX input files into commands understood by TeX or **troff**
pic2graph	Converts a PIC diagram into a cropped image
post-grohtml	Translates the output of GNU **troff** to HTML
pre-grohtml	Translates the output of GNU **troff** to HTML

refer	Copies the contents of a file to the standard output, except that lines between *.[* and *.]* are interpreted as citations, and lines between *.R1* and *.R2* are interpreted as commands for how citations are to be processed
soelim	Reads files and replaces lines of the form *.so file* by the contents of the mentioned *file*
tbl	Compiles descriptions of tables embedded within troff input files into commands that are understood by **troff**
tfmtodit	Creates a font file for use with **groff -Tdvi**
troff	Is highly compatible with Unix **troff**; it should usually be invoked using the **groff** command, which will also run preprocessors and post-processors in the appropriate order and with the appropriate options

6.41. Gzip-1.3.12

The Gzip package contains programs for compressing and decompressing files.

Approximate build time: less than 0.1 SBU
Required disk space: 2.2 MB

6.41.1. Installation of Gzip

The version of the function “futimens” used by Gzip is incompatible with the version that current Glibc provides, so we'll rename the function:

```
sed -i 's/futimens/gl_&/' gzip.c lib/utimens.{c,h}
```

Prepare Gzip for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Move some programs that do not need to be on the root filesystem:

```
mv -v /bin/{gzexe,uncompress,zcmp,zdiff,zegrep} /usr/bin
mv -v /bin/{zfgrep,zforce,zgrep,zless,zmore,znew} /usr/bin
```

6.41.2. Contents of Gzip

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

Short Descriptions

gunzip	Decompresses gzipped files
gzexe	Creates self-decompressing executable files
gzip	Compresses the given files using Lempel-Ziv (LZ77) coding
uncompress	Decompresses compressed files
zcat	Decompresses the given gzipped files to standard output
zcmp	Runs **cmp** on gzipped files
zdiff	Runs **diff** on gzipped files
zegrep	Runs **egrep** on gzipped files
zfgrep	Runs **fgrep** on gzipped files

zforce	Forces a `.gz` extension on all given files that are gzipped files, so that **gzip** will not compress them again; this can be useful when file names were truncated during a file transfer
zgrep	Runs **grep** on gzipped files
zless	Runs **less** on gzipped files
zmore	Runs **more** on gzipped files
znew	Re-compresses files from **compress** format to **gzip** format—`.Z` to `.gz`

6.42. Inetutils-1.5

The Inetutils package contains programs for basic networking.

Approximate build time: 0.3 SBU
Required disk space: 12 MB

6.42.1. Installation of Inetutils

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

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

Inetutils has a minor issue with GCC-4.3.2. Fix it by issuing the following command:

```
sed -i 's@<sys/types.h>@<sys/types.h>\n#include <stdlib.h>@' \
    libicmp/icmp_timestamp.c
```

Prepare Inetutils for compilation:

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

The meaning of the configure options:

--disable-ifconfig
: This option prevents Inetutils from installing the **ifconfig** program, which can be used to configure network interfaces. LFS uses **ip** from IPRoute2 to perform this task.

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

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

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

--disable-servers
: This disables the installation of the various network servers included as part of the Inetutils package. These servers are deemed not appropriate in a basic LFS system. Some are insecure by nature and are only considered safe on trusted networks. More information can be found at *http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html*. Note that better replacements are available for many of these servers.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move the **ping** program to its FHS-compliant place:

```
mv -v /usr/bin/ping /bin
```

6.42.2. Contents of Inetutils

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

Short Descriptions

ftp	Is the file transfer protocol program
ping	Sends echo-request packets and reports how long the replies take
ping6	A version of **ping** for IPv6 networks
rcp	Performs remote file copy
rlogin	Performs remote login
rsh	Runs a remote shell
talk	Is used to chat with another user
telnet	An interface to the TELNET protocol
tftp	A trivial file transfer program

6.43. IPRoute2-2.6.26

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

Approximate build time: 0.2 SBU
Required disk space: 5.6 MB

6.43.1. Installation of IPRoute2

Compile the package:

```
make DESTDIR= SBINDIR=/sbin
```

The meaning of the make options:

DESTDIR=
: This ensures that the IPRoute2 binaries will install into the correct directory. By default, *DESTDIR* is set to `/usr`.

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

This package comes with a testsuite, but due to assumptions it makes, it is not possible to reliably run these tests from within the chroot environment. If you wish to run these tests after booting into your new LFS system, ensure you select `/proc/config.gz` CONFIG_IKCONFIG_PROC ("General setup" -> "Enable access to .config through /proc/config.gz") support into your kernel then run 'make alltests' from the `testsuite/` subdirectory.

Install the package:

```
make DESTDIR= SBINDIR=/sbin MANDIR=/usr/share/man \
     DOCDIR=/usr/share/doc/iproute2-2.6.26 install
```

The **arpd** binary links against the Berkeley DB libraries that reside in `/usr` and uses a database in `/var/lib/arpd/arpd.db`. Thus, according to the FHS, it must be in `/usr/sbin`. Move it there:

```
mv -v /sbin/arpd /usr/sbin
```

6.43.2. Contents of IPRoute2

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

Short Descriptions

arpd Userspace ARP daemon, useful in really large networks, where the kernelspace ARP implementation is insufficient, or when setting up a honeypot

ctstat Connection status utility

genl

ifcfg A shell script wrapper for the **ip** command. Note that it requires the **arping** and **rdisk** programs from the iputils package found at *http://www.skbuff.net/iputils/*.

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

ip The main executable. It has several different functions:

ip link ***<device>*** allows users to look at the state of devices and to make changes

ip addr allows users to look at addresses and their properties, add new addresses, and delete old ones

ip neighbor allows users to look at neighbor bindings and their properties, add new neighbor entries, and delete old ones

ip rule allows users to look at the routing policies and change them

ip route allows users to look at the routing table and change routing table rules

ip tunnel allows users to look at the IP tunnels and their properties, and change them

ip maddr allows users to look at the multicast addresses and their properties, and change them

ip mroute allows users to set, change, or delete the multicast routing

ip monitor allows users to continously monitor the state of devices, addresses and routes

lnstat Provides Linux network statistics. It is a generalized and more feature-complete replacement for the old **rtstat** program

nstat Shows network statistics

routef A component of **ip route**. This is for flushing the routing tables

routel A component of **ip route**. This is for listing the routing tables

rtacct Displays the contents of `/proc/net/rt_acct`

rtmon Route monitoring utility

rtpr Converts the output of **ip -o** back into a readable form

rtstat Route status utility

ss Similar to the **netstat** command; shows active connections

tc Traffic Controlling Executable; this is for Quality Of Service (QOS) and Class Of Service (COS) implementations

tc qdisc allows users to setup the queueing discipline

tc class allows users to setup classes based on the queuing discipline scheduling

tc estimator allows users to estimate the network flow into a network

tc filter allows users to setup the QOS/COS packet filtering

tc policy allows users to setup the QOS/COS policies

6.44. Kbd-1.14.1

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

Approximate build time: less than 0.1 SBU
Required disk space: 12.5 MB

6.44.1. Installation of Kbd

The behaviour of the Backspace and Delete keys is not consistent across the keymaps in the Kbd package. The following patch fixes this issue for i386 keymaps:

```
patch -Np1 -i ../kbd-1.14.1-backspace-1.patch
```

After patching, the Backspace key generates the character with code 127, and the Delete key generates a well-known escape sequence.

In this version of Kbd the instructions to build `getkeycodes`, `setkeycodes` and `resizecons` do not get passed over to the generated `Makefile` as they should. So that these programs are built and installed add two lines to the top of `src/Makefile.in`:

```
sed -i -e '1i KEYCODES_PROGS = @KEYCODES_PROGS@' \
    -e '1i RESIZECONS_PROGS = @RESIZECONS_PROGS@' src/Makefile.in
```

This version of Kbd will also install man pages for optional programs even if we don't use the *--enable-optional-progs* option to build them. Fix this behavior:

```
var=OPTIONAL_PROGS
sed -i "s/ifdef $var/ifeq (\$($var),yes)/" man/Makefile.in
unset var
```

Prepare Kbd for compilation:

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

The meaning of the configure options:

--datadir=/lib/kbd
: This option puts keyboard layout data in a directory that will always be on the root partition instead of the default `/usr/share/kbd`.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Note

For some languages (e.g., Belarusian) the Kbd package doesn't provide a useful keymap where the stock “by” keymap assumes the ISO-8859-5 encoding, and the CP1251 keymap is normally used. Users of such languages have to download working keymaps separately.

Some of the scripts in the LFS-Bootscripts package depend on **kbd_mode**, **loadkeys**, **openvt**, and **setfont**. As `/usr` may not be available during the early stages of booting, those binaries need to be on the root partition:

```
mv -v /usr/bin/{kbd_mode,loadkeys,openvt,setfont} /bin
```

If desired, install the documentation:

```
mkdir -v /usr/share/doc/kbd-1.14.1
cp -R -v doc/* \
         /usr/share/doc/kbd-1.14.1
```

6.44.2. Contents of Kbd

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

Short Descriptions

chvt	Changes the foreground virtual terminal
deallocvt	Deallocates unused virtual terminals
dumpkeys	Dumps the keyboard translation tables
fgconsole	Prints the number of the active virtual terminal
getkeycodes	Prints the kernel scancode-to-keycode mapping table
kbd_mode	Reports or sets the keyboard mode
kbdrate	Sets the keyboard repeat and delay rates
loadkeys	Loads the keyboard translation tables
loadunimap	Loads the kernel unicode-to-font mapping table
mapscrn	An obsolete program that used to load a user-defined output character mapping table into the console driver; this is now done by **setfont**
openvt	Starts a program on a new virtual terminal (VT)
psfaddtable	A link to **psfxtable**
psfgettable	A link to **psfxtable**
psfstriptable	A link to **psfxtable**
psfxtable	Handle Unicode character tables for console fonts
resizecons	Changes the kernel idea of the console size
setfont	Changes the Enhanced Graphic Adapter (EGA) and Video Graphics Array (VGA) fonts on the console
setkeycodes	Loads kernel scancode-to-keycode mapping table entries; this is useful if there are unusual keys on the keyboard
setleds	Sets the keyboard flags and Light Emitting Diodes (LEDs)
setmetamode	Defines the keyboard meta-key handling

showconsolefont	Shows the current EGA/VGA console screen font
showkey	Reports the scancodes, keycodes, and ASCII codes of the keys pressed on the keyboard
unicode_start	Puts the keyboard and console in UNICODE mode. Don't use this program unless your keymap file is in the ISO-8859-1 encoding. For other encodings, this utility produces incorrect results.
unicode_stop	Reverts keyboard and console from UNICODE mode

6.45. Less-418

The Less package contains a text file viewer.

Approximate build time: 0.1 SBU
Required disk space: 2.8 MB

6.45.1. Installation of Less

Prepare Less for compilation:

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

The meaning of the configure options:

--sysconfdir=/etc
This option tells the programs created by the package to look in `/etc` for the configuration files.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.45.2. Contents of Less

Installed programs: less, lessecho, and lesskey

Short Descriptions

less A file viewer or pager; it displays the contents of the given file, letting the user scroll, find strings, and jump to marks

lessecho Needed to expand meta-characters, such as * and ?, in filenames on Unix systems

lesskey Used to specify the key bindings for **less**

6.46. Make-3.81

The Make package contains a program for compiling packages.

Approximate build time: 0.1 SBU
Required disk space: 9.6 MB

6.46.1. Installation of Make

Prepare Make for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

6.46.2. Contents of Make

Installed program: make

Short Descriptions

make Automatically determines which pieces of a package need to be (re)compiled and then issues the relevant commands

6.47. Man-DB-2.5.2

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

Approximate build time: 0.3 SBU
Required disk space: 20 MB

6.47.1. Installation of Man-DB

LFS creates `/usr/man` and `/usr/local/man` as symlinks. Remove them from the `man_db.conf` file to prevent redundant results when using programs such as **whatis**:

```
sed -i -e '\%\t/usr/man%d' -e '\%\t/usr/local/man%d' src/man_db.conf.in
```

Prepare Man-DB for compilation:

```
./configure --prefix=/usr --libexecdir=/usr/lib \
    --sysconfdir=/etc --disable-setuid \
    --enable-mb-groff --with-browser=/usr/bin/lynx \
    --with-col=/usr/bin/col --with-vgrind=/usr/bin/vgrind \
    --with-grap=/usr/bin/grap
```

The meaning of the configure options:

--disable-setuid
: This disables making the **man** program setuid to user `man`.

--enable-mb-groff
: This switch tells man-db to expect the Debian multibyte patched version of groff.

--with-...
: These four parameters are used to set some default programs. The **col** program is a part of the Util-linux-ng package, **lynx** is a text-based web browser (see BLFS for installation instructions), **vgrind** converts program sources to Groff input, and **grap** is useful for typesetting graphs in Groff documents. The **vgrind** and **grap** programs are not normally needed for viewing manual pages. They are not part of LFS or BLFS, but you should be able to install them yourself after finishing LFS if you wish to do so.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.47.2. Non-English Manual Pages in LFS

Some packages provide non-English manual pages. They are displayed correctly only if their location and encoding matches the expectation of the "man" program. However, different Linux distributions have different policies (expressed in the choice of the **man** program, its configuration and patches applied to it) concerning the character encoding in which manual pages are stored in the filesystem.

E.g., Debian previously required Russian manual pages to be encoded in KOI8-R and to be placed in `/usr/share/man/ru`. Now, in addition, their **man** program (Man-DB) searches for UTF-8 encoded Russian manual pages in `/usr/share/man/ru.UTF-8`. On the other hand, Fedora uses UTF-8 encoded manual pages exclusively. Russian manual pages are found in `/usr/share/man/ru` and their **man** program doesn't acknowledge `/usr/share/man/ru.UTF-8`. Many other distributions ignore the on disk encodings completely, leaving the end user with a mix of improperly encoded manual pages for their configuration. When **man** processes the requtested page, it will display the contents as configured, resulting in completely unreadable text if the on disk encoding is not what is expected for that configuration.

Disagreement about the expected encoding of manual pages amongst distribution vendors, has led to confusion for upstream package maintainers. One package may contain UTF-8 manual pages, while another ships with manual pages in legacy encodings. **man** searches for manual pages based on the user's locale settings. Man-DB uses a built-in table (see below) to determine the on disk encoding of manual pages found for a user's locale, only if the directories found do not have an extension that describes the encoding. E.g., because of ".UTF-8" in the directory name, Man-DB knows that all manual pages residing in `/usr/share/man/fr.UTF-8` are UTF-8 encoded and, according to the built-in table, expects all manual pages residing in `/usr/share/man/ru` to be encoded using KOI8-R.

Table 6.1. Expected character encoding of legacy 8-bit manual pages

Language (code)	Encoding	Language (code)	Encoding
Danish (da)	ISO-8859-1	Bulgarian (bg)	CP1251
German (de)	ISO-8859-1	Czech (cs)	ISO-8859-2
English (en)	ISO-8859-1	Croatian (hr)	ISO-8859-2
Spanish (es)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
Finnish (fi)	ISO-8859-1	Japanese (ja)	EUC-JP
French (fr)	ISO-8859-1	Korean (ko)	EUC-KR
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Russian (ru)	KOI8-R
Indonesian (id)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Icelandic (is)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Italian (it)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Dutch (nl)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Norwegian (no)	ISO-8859-1	Simplified Chinese, Singapore (zh_SG)	GBK
Portuguese (pt)	ISO-8859-1	Traditional Chinese (zh_TW)	BIG5
Swedish (sv)	ISO-8859-1	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS

Note

Manual pages in languages not in the list are not supported. Norwegian does not work because of the transition from no_NO to nb_NO locale, and will be fixed in the next release of Man-DB. Korean is currently non functional because of incomplete fixes in the Debian Groff patch applied in LFS.

Packages may install manual pages into an improperly named directory, depending on which distributions the author develops the package for. To assist in the conversion of the manual pages to the proper encoding for the directory in which they are installed, the **convert-mans** script was written. It will convert manual pages to another encoding before (or after) installation. Install the **convert-mans** script with the following instructions:

```
cat >> convert-mans << "EOF"
#!/bin/sh -e
FROM="$1"
TO="$2"
shift ; shift
while [ $# -gt 0 ]
do
        FILE="$1"
        shift
        iconv -f "$FROM" -t "$TO" "$FILE" >.tmp.iconv
        mv .tmp.iconv "$FILE"
done
EOF
install -v -m755 convert-mans  /usr/bin
```

If upstream distributes the manual pages in a legacy encoding, the manual pages can simply be copied to /usr/share/man/<language code>. For example, *German manual pages* can be installed with the following commands:

```
mkdir -p /usr/share/man/de
cp -rv man? /usr/share/man/de
```

If upstream distributes manual pages in UTF-8 (i.e., “for RedHat”) instead of the encoding listed in the table above, they can either be converted from UTF-8 to the encoding listed in the table above, or they can be installed directly into /usr/share/man/<language code>.UTF-8.

For example, to install *French manual pages* in the legacy encoding, use the following commands:

```
convert-mans UTF-8 ISO-8859-1 man?/*.?
mkdir -p /usr/share/man/fr
cp -rv man? /usr/share/man/fr
```

Note

The French manual pages ship with ready made scripts to do the same conversion. The above instructions are used only as an example for use of the **convert-mans** script.

Finally, as an example installation of UTF-8 manual pages, again, the French manual pages could be installed with the following commands:

```
mkdir -p /usr/share/man/fr.UTF-8
cp -rv man? /usr/share/man/fr.UTF-8
```

6.47.3. Contents of Man-DB

Installed programs: apropos, catman, convert-mans, lexgrog, man, mandb, manpath, whatis, and zsoelim

Short Descriptions

apropos	Searches the **whatis** database and displays the short descriptions of system commands that contain a given string
catman	Creates or updates the pre-formatted manual pages
convert-mans	Reformats manual pages into the chosen encoding.
lexgrog	Displays one-line summary information about a given manual page
man	Formats and displays the requested manual page
mandb	Creates or updates the **whatis** database
manpath	Displays the contents of $MANPATH or (if $MANPATH is not set) a suitable search path based on the settings in man.conf and the user's environment
whatis	Searches the **whatis** database and displays the short descriptions of system commands that contain the given keyword as a separate word
zsoelim	Reads files and replaces lines of the form *.so file* by the contents of the mentioned *file*

6.48. Module-Init-Tools-3.4.1

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

Approximate build time: less than 0.1 SBU
Required disk space: 8 MB

6.48.1. Installation of Module-Init-Tools

The tarball only contains sgml source for the manpages. The following patch contains the result of processing this through **docbook2man** (see *http://www.linuxfromscratch.org/blfs/view/svn/pst/docbook-utils.html*) which we do not build as part of a basic LFS installation:

```
patch -Np1 -i ../module-init-tools-3.4.1-manpages-1.patch
```

The testsuite of this package is geared towards the needs of its Maintainer. The command **make check** builds a specially wrapped version of modprobe which is useless for normal operation. To run this (about 0.2 SBU), issue the following commands (note that the **make clean** command is required to clean up the source tree before recompiling for normal use):

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

Prepare Module-Init-Tools for compilation:

```
./configure --prefix=/ --enable-zlib --mandir=/usr/share/man
```

Compile the package:

```
make
```

Install the package:

```
make INSTALL=install install
```

The meaning of the make parameter:

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

6.48.2. Contents of Module-Init-Tools

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

Short Descriptions

depmod Creates a dependency file based on the symbols it finds in the existing set of modules; this dependency file is used by **modprobe** to automatically load the required modules

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

6.49. Patch-2.5.4

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

Approximate build time: less than 0.1 SBU
Required disk space: 1.6 MB

6.49.1. Installation of Patch

Prepare Patch for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.49.2. Contents of Patch

Installed program: patch

Short Descriptions

patch Modifies files according to a patch file. A patch file is normally a difference listing created with the **diff** program. By applying these differences to the original files, **patch** creates the patched versions.

6.50. Psmisc-22.6

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

Approximate build time: less than 0.1 SBU
Required disk space: 2.2 MB

6.50.1. Installation of Psmisc

Prepare Psmisc for compilation:

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

The meaning of the configure options:

--exec-prefix=""
This ensures that the Psmisc binaries will install into `/bin` instead of `/usr/bin`. This is the correct location according to the FHS, because some of the Psmisc binaries are used by the LFS-Bootscripts package.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

There is no reason for the **pstree** and **pstree.x11** programs to reside in `/bin`. Therefore, move them to `/usr/bin`:

```
mv -v /bin/pstree* /usr/bin
```

By default, Psmisc's **pidof** program is not installed. This usually is not a problem because it is installed later in the Sysvinit package, which provides a better **pidof** program. If Sysvinit will not be used for a particular system, complete the installation of Psmisc by creating the following symlink:

```
ln -sv killall /bin/pidof
```

6.50.2. Contents of Psmisc

Installed programs: fuser, killall, oldfuser, peekfd, pstree, and pstree.x11 (link to pstree)

Short Descriptions

fuser	Reports the Process IDs (PIDs) of processes that use the given files or file systems
killall	Kills processes by name; it sends a signal to all processes running any of the given commands
oldfuser	Reports the Process IDs (PIDs) of processes that use the given files or file systems
peekfd	Peek at file descriptors of a running process, given its PID
pstree	Displays running processes as a tree
pstree.x11	Same as **pstree**, except that it waits for confirmation before exiting

6.51. Shadow-4.1.2.1

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

Approximate build time: 0.3 SBU
Required disk space: 28 MB

6.51.1. Installation of Shadow

> **Note**
>
> If you would like to enforce the use of strong passwords, refer to *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* for installing CrackLib prior to building Shadow. Then add *`--with-libcrack`* to the **configure** command below.

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

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

Disable the installation of Chinese and Korean manual pages, since Man-DB cannot format them properly:

```
sed -i -e 's/ ko//' -e 's/ zh_CN zh_TW//' man/Makefile.in
```

Shadow supplies other manual pages in a UTF-8 encoding. Man-DB can display these in the recommended encodings by using the **convert-mans** script which was installed during the Man-DB package:

```
for i in de es fi fr id it pt_BR; do
    convert-mans UTF-8 ISO-8859-1 man/${i}/*.?
done

for i in cs hu pl; do
    convert-mans UTF-8 ISO-8859-2 man/${i}/*.?
done

convert-mans UTF-8 EUC-JP man/ja/*.?
convert-mans UTF-8 KOI8-R man/ru/*.?
convert-mans UTF-8 ISO-8859-9 man/tr/*.?
```

Instead of using the default *crypt* method, use the more secure *MD5* method of password encryption, which also allows passwords longer than 8 characters. It is also necessary to change the obsolete `/var/spool/mail` location for user mailboxes that Shadow uses by default to the `/var/mail` location used currently:

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD MD5@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

> **Note**
>
> If you chose to build Shadow with Cracklib support, run the following:
>
> ```
> sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
>     etc/login.defs
> ```

Prepare Shadow for compilation:

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

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Move a misplaced program to its proper location:

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

6.51.2. Configuring Shadow

This package contains utilities to add, modify, and delete users and groups; set and change their passwords; and perform other administrative tasks. For a full explanation of what *password shadowing* means, see the `doc/HOWTO` file within the unpacked source tree. If using Shadow support, keep in mind that programs which need to verify passwords (display managers, FTP programs, pop3 daemons, etc.) must be Shadow-compliant. That is, they need to be able to work with shadowed passwords.

To enable shadowed passwords, run the following command:

```
pwconv
```

To enable shadowed group passwords, run:

```
grpconv
```

Shadow's stock configuration for the **useradd** utility has a few caveats that need some explanation. First, the default action for the **useradd** utility is to create the user and a group of the same name as the user. By default the user ID (UID) and group ID (GID) numbers will begin with 1000. This means if you don't pass parameters to **useradd**, each user will be a member of a unique group on the system. If this behaviour is undesireable, you'll need to pass the *`-g`* parameter to **useradd**. The default parameters are stored in the `/etc/default/useradd` file. You may need to modify two parameters in this file to suit your particular needs.

`/etc/default/useradd` Parameter Explanations

`GROUP=1000`

This parameter sets the beginning of the group numbers used in the /etc/group file. You can modify it to anything you desire. Note that **useradd** will never reuse a UID or GID. If the number identified in this parameter is used, it will use the next available number after this. Note also that if you don't have a group 1000 on your system the first time you use **useradd** without the *`-g`* parameter, you'll get a message displayed on the terminal that says: `useradd: unknown GID 1000`. You may disregard this message and group number 1000 will be used.

`CREATE_MAIL_SPOOL=yes`

This parameter causes **useradd** to create a mailbox file for the newly created user. **useradd** will make the group ownership of this file to the `mail` group with 0660 permissions. If you would prefer that these mailbox files are not created by **useradd**, issue the following command:

```
sed -i 's/yes/no/' /etc/default/useradd
```

6.51.3. Setting the root password

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

```
passwd root
```

6.51.4. Contents of Shadow

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

Short Descriptions

chage	Used to change the maximum number of days between obligatory password changes
chfn	Used to change a user's full name and other information
chgpasswd	Used to update group passwords in batch mode
chpasswd	Used to update user passwords in batch mode
chsh	Used to change a user's default login shell
expiry	Checks and enforces the current password expiration policy
faillog	Is used to examine the log of login failures, to set a maximum number of failures before an account is blocked, or to reset the failure count
gpasswd	Is used to add and delete members and administrators to groups
groupadd	Creates a group with the given name
groupdel	Deletes the group with the given name
groupmems	Allows a user to administer his/her own group membership list without the requirement of super user privileges.
groupmod	Is used to modify the given group's name or GID
grpck	Verifies the integrity of the group files `/etc/group` and `/etc/gshadow`
grpconv	Creates or updates the shadow group file from the normal group file
grpunconv	Updates `/etc/group` from `/etc/gshadow` and then deletes the latter
lastlog	Reports the most recent login of all users or of a given user
login	Is used by the system to let users sign on
logoutd	Is a daemon used to enforce restrictions on log-on time and ports
newgrp	Is used to change the current GID during a login session
newusers	Is used to create or update an entire series of user accounts
nologin	Displays a message that an account is not available. Designed to be used as the default shell for accounts that have been disabled
passwd	Is used to change the password for a user or group account
pwck	Verifies the integrity of the password files `/etc/passwd` and `/etc/shadow`

pwconv	Creates or updates the shadow password file from the normal password file
pwunconv	Updates `/etc/passwd` from `/etc/shadow` and then deletes the latter
sg	Executes a given command while the user's GID is set to that of the given group
su	Runs a shell with substitute user and group IDs
useradd	Creates a new user with the given name, or updates the default new-user information
userdel	Deletes the given user account
usermod	Is used to modify the given user's login name, User Identification (UID), shell, initial group, home directory, etc.
vigr	Edits the `/etc/group` or `/etc/gshadow` files
vipw	Edits the `/etc/passwd` or `/etc/shadow` files

6.52. Sysklogd-1.5

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

Approximate build time: less than 0.1 SBU
Required disk space: 0.6 MB

6.52.1. Installation of Sysklogd

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.52.2. Configuring Sysklogd

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

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

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

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

6.52.3. Contents of Sysklogd

Installed programs: klogd and syslogd

Short Descriptions

klogd A system daemon for intercepting and logging kernel messages

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

6.53. Sysvinit-2.86

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

Approximate build time:	less than 0.1 SBU
Required disk space:	1 MB

6.53.1. Installation of Sysvinit

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

```
sed -i 's@Sending processes@& configured via /etc/inittab@g' \
    src/init.c
```

A maintained version of the **wall** program is installed later on during the Util-linux-ng installation. Suppress the installation of this program and its man page:

```
sed -i -e 's/utmpdump wall/utmpdump/' \
       -e 's/mountpoint.1 wall.1/mountpoint.1/' src/Makefile
```

Compile the package:

```
make -C src
```

This package does not come with a test suite.

Install the package:

```
make -C src install
```

6.53.2. Configuring Sysvinit

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

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

id:3:initdefault:

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

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

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

su:S016:once:/sbin/sulogin

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
```

6.53.3. Contents of Sysvinit

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

Short Descriptions

bootlogd Logs boot messages to a log file

halt Normally invokes **shutdown** with the *-h* option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file `/var/log/wtmp` that the system is being brought down

init The first process to be started when the kernel has initialized the hardware which takes over the boot process and starts all the proceses it is instructed to

killall5 Sends a signal to all processes, except the processes in its own session so it will not kill the shell running the script that called it

last	Shows which users last logged in (and out), searching back through the `/var/log/wtmp` file; it also shows system boots, shutdowns, and run-level changes
lastb	Shows the failed login attempts, as logged in `/var/log/btmp`
mesg	Controls whether other users can send messages to the current user's terminal
mountpoint	Checks if the directory is a mountpoint
pidof	Reports the PIDs of the given programs
poweroff	Tells the kernel to halt the system and switch off the computer (see **halt**)
reboot	Tells the kernel to reboot the system (see **halt**)
runlevel	Reports the previous and the current run-level, as noted in the last run-level record in `/var/run/utmp`
shutdown	Brings the system down in a secure way, signaling all processes and notifying all logged-in users
sulogin	Allows `root` to log in; it is normally invoked by **init** when the system goes into single user mode
telinit	Tells **init** which run-level to change to
utmpdump	Displays the content of the given login file in a more user-friendly format

6.54. Tar-1.20

The Tar package contains an archiving program.

Approximate build time: 0.3 SBU
Required disk space: 19.9 MB

6.54.1. Installation of Tar

Prepare Tar for compilation:

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

Compile the package:

```
make
```

To test the results (about 1 SBU), issue:

```
make check
```

Install the package:

```
make install
```

6.54.2. Contents of Tar

Installed programs: rmt and tar

Short Descriptions

rmt Remotely manipulates a magnetic tape drive through an interprocess communication connection

tar Creates, extracts files from, and lists the contents of archives, also known as tarballs

6.55. Texinfo-4.13a

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

Approximate build time: 0.3 SBU
Required disk space: 20 MB

6.55.1. Installation of Texinfo

Prepare Texinfo for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

Optionally, install the components belonging in a TeX installation:

```
make TEXMF=/usr/share/texmf install-tex
```

The meaning of the make parameter:

`TEXMF=/usr/share/texmf`
: The `TEXMF` makefile variable holds the location of the root of the TeX tree if, for example, a TeX package will be installed later.

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

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

6.55.2. Contents of Texinfo

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

Short Descriptions

info Used to read info pages which are similar to man pages, but often go much deeper than just explaining all the available command line options. For example, compare **man bison** and **info bison**.

infokey	Compiles a source file containing Info customizations into a binary format
install-info	Used to install info pages; it updates entries in the **info** index file
makeinfo	Translates the given Texinfo source documents into info pages, plain text, or HTML
texi2dvi	Used to format the given Texinfo document into a device-independent file that can be printed
texi2pdf	Used to format the given Texinfo document into a Portable Document Format (PDF) file
texindex	Used to sort Texinfo index files

6.56. Udev-130

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

Approximate build time: 0.2 SBU
Required disk space: 10 MB

6.56.1. Installation of Udev

The udev-config tarball contains LFS-specific files used to configure Udev. Unpack it into the Udev source directory:

```
tar -xvf ../udev-config-20081015.tar.bz2
```

Create some devices and directories that Udev cannot handle due to them being required very early in the boot process, or by Udev itself:

```
install -dv /lib/{firmware,udev/devices/{pts,shm}}
mknod -m0666 /lib/udev/devices/null c 1 3
mknod -m0600 /lib/udev/devices/kmsg c 1 11
ln -sv /proc/self/fd /lib/udev/devices/fd
ln -sv /proc/self/fd/0 /lib/udev/devices/stdin
ln -sv /proc/self/fd/1 /lib/udev/devices/stdout
ln -sv /proc/self/fd/2 /lib/udev/devices/stderr
ln -sv /proc/kcore /lib/udev/devices/core
```

Prepare the package for compilation:

```
./configure --prefix=/usr \
            --exec-prefix= \
            --sysconfdir=/etc
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

Udev has to be configured in order to work properly, as its default configuration does not cover all devices. First install two extra rules files from Udev to help support device-mapper and RAID setups:

```
install -m644 -v rules/packages/64-*.rules \
    /lib/udev/rules.d/
```

Now install a file to create symlinks for certain hand-held devices:

```
install -m644 -v rules/packages/40-pilot-links.rules \
    /lib/udev/rules.d/
```

Now install the LFS-specific custom rules files:

```
cd udev-config-20081015
make install
```

Install the documentation that explains the LFS-specific rules files:

```
make install-doc
```

Install the documentation that explains the commonly-used rules files provided by Udev:

```
make install-extra-doc
```

Install the documentation that explains how to create custom Udev rules:

```
cd ..
install -m644 -v -D docs/writing_udev_rules/index.html \
    /usr/share/doc/udev-130/index.html
```

6.56.2. Contents of Udev

Installed programs:	ata_id, cdrom_id, collect, create_floppy_devices, edd_id, firmware.sh, fstab_import, path_id, scsi_id, udevadm, udevd, usb_id, vol_id, write_cd_rules, and write_net_rules
Installed libraries:	libudev and libvolume_id
Installed directory:	/etc/udev

Short Descriptions

ata_id	Provides Udev with a unique string and additional information (uuid, label) for an ATA drive
cdrom_id	Provides Udev with the capabilities of a CD-ROM or DVD-ROM drive
collect	Given an ID for the current uevent and a list of IDs (for all target uevents), registers the current ID and indicates whether all target IDs have been registered
create_floppy_devices	Creates all possible floppy devices based on the CMOS type
edd_id	Provides Udev with the EDD ID for a BIOS disk drive
firmware.sh	Uploads firmware to devices
fstab_import	Finds an entry in `/etc/fstab` that matches the current device, and provides its information to Udev
path_id	Provides the shortest possible unique hardware path to a device
scsi_id	Provides Udev with a unique SCSI identifier based on the data returned from sending a SCSI INQUIRY command to the specified device
udevadm	Generic udev administration tool: controls the udevd daemon, provides info from the Udev database, monitors uevents, waits for uevents to finish, tests Udev configuration, and triggers uevents for a given device
udevd	A daemon that listens for uevents on the netlink socket, creates devices and runs the configured external programs in response to these uevents
usb_id	Provides Udev with information about USB devices

vol_id	Provides Udev with the label and uuid of a filesystem
write_cd_rules	A script which generates Udev rules to provide stable names for optical drives (see also Section 7.12, “Creating Custom Symlinks to Devices”)
write_net_rules	A script which generates rules to provide stable names for network interfaces (see also Section 7.13, “Configuring the network Script”)
libudev	A library interface to udev device information
libvolume_id	A library interface to read volume labels and uuids
`/etc/udev`	Contains Udev configuration files, device permissions, and rules for device naming

6.57. Util-linux-ng-2.14.1

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

Approximate build time: 0.3 SBU
Required disk space: 29 MB

6.57.1. FHS compliance notes

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

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

6.57.2. Installation of Util-linux-ng

```
./configure --enable-arch --enable-partx --enable-write
```

The meaning of the configure options:

--enable-arch
Enables building the **arch** program

--enable-partx
Enables building the **addpart**, **delpart** and **partx** programs

--enable-write
Enables building the **write** program

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
```

6.57.3. Contents of Util-linux-ng

Installed programs: addpart, agetty, arch, blockdev, cal, cfdisk, chkdupexe, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, delpart, dmesg, fdformat, fdisk, flock, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, i386, ionice, ipcrm, ipcs, isosize, ldattach, line, linux32, linux64, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, partx, pg, pivot_root, readprofile, rename, renice, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, swapon, tailf, taskset, tunelp, ul, umount, wall, whereis, and write

Short Descriptions

addpart Informs the Linux kernel of new partitions

agetty	Opens a tty port, prompts for a login name, and then invokes the **login** program
arch	Reports the machine's architecture
blockdev	Allows users to call block device ioctls from the command line
cal	Displays a simple calendar
cfdisk	Manipulates the partition table of the given device
chkdupexe	Finds duplicate executables
chrt	Manipulates real-time attributes of a process
col	Filters out reverse line feeds
colcrt	Filters **nroff** output for terminals that lack some capabilities, such as overstriking and half-lines
colrm	Filters out the given columns
column	Formats a given file into multiple columns
ctrlaltdel	Sets the function of the Ctrl+Alt+Del key combination to a hard or a soft reset
cytune	Tunes the parameters of the serial line drivers for Cyclades cards
ddate	Gives the Discordian date or converts the given Gregorian date to a Discordian one
delpart	Asks the Linux kernel to remove a partition
dmesg	Dumps the kernel boot messages
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the paritition table of the given device
flock	Acquires a file lock and then executes a command with the lock held
fsck.cramfs	Performs a consistency check on the Cramfs file system on the given device
fsck.minix	Performs a consistency check on the Minix file system on the given device
getopt	Parses options in the given command line
hexdump	Dumps the given file in hexadecimal or in another given format
hwclock	Reads or sets the system's hardware clock, also called the Real-Time Clock (RTC) or Basic Input-Output System (BIOS) clock
i386	A symbolic link to setarch
ionice	Gets or sets the io scheduling class and priority for a program
ipcrm	Removes the given Inter-Process Communication (IPC) resource
ipcs	Provides IPC status information
isosize	Reports the size of an iso9660 file system
ldattach	Attaches a line discipline to a serial line
linux32	A symbolic link to setarch
linux64	A symbolic link to setarch
line	Copies a single line
logger	Enters the given message into the system log
look	Displays lines that begin with the given string

losetup	Sets up and controls loop devices
mcookie	Generates magic cookies (128-bit random hexadecimal numbers) for **xauth**
mkfs	Builds a file system on a device (usually a hard disk partition)
mkfs.bfs	Creates a Santa Cruz Operations (SCO) bfs file system
mkfs.cramfs	Creates a cramfs file system
mkfs.minix	Creates a Minix file system
mkswap	Initializes the given device or file to be used as a swap area
more	A filter for paging through text one screen at a time
mount	Attaches the file system on the given device to a specified directory in the file-system tree
namei	Shows the symbolic links in the given pathnames
partx	Tells the kernel about the presence and numbering of on-disk partitions
pg	Displays a text file one screen full at a time
pivot_root	Makes the given file system the new root file system of the current process
readprofile	Reads kernel profiling information
rename	Renames the given files, replacing a given string with another
renice	Alters the priority of running processes
rev	Reverses the lines of a given file
rtcwake	Used to enter a system sleep state until specified wakeup time
script	Makes a typescript of a terminal session
scriptreplay	Plays back typescripts using timing information
setarch	Changes reported architecture in a new program environment and sets personality flags
setsid	Runs the given program in a new session
setterm	Sets terminal attributes
sfdisk	A disk partition table manipulator
swapon	Enables devices and files for paging and swapping and lists the devices and files currently in use
tailf	Tracks the growth of a log file. Displays the last 10 lines of a log file, then continues displaying any new entries in the log file as they are created
taskset	Retrieves or sets a process' CPU affinity
tunelp	Tunes the parameters of the line printer
ul	A filter for translating underscores into escape sequences indicating underlining for the terminal in use
umount	Disconnects a file system from the system's file tree
wall	Displays the contents of a file or, by default, its standard input, on the terminals of all currently logged in users
whereis	Reports the location of the binary, source, and man page for the given command
write	Sends a message to the given user *if* that user has not disabled receipt of such messages

6.58. Vim-7.2

The Vim package contains a powerful text editor.

Approximate build time: 0.8 SBU
Required disk space: 67 MB

Alternatives to Vim

If you prefer another editor—such as Emacs, Joe, or Nano—please refer to *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html* for suggested installation instructions.

6.58.1. Installation of Vim

First, unpack both `vim-7.2.tar.bz2` and (optionally) `vim-7.2-lang.tar.gz` archives into the same directory.

Apply a patch which fixes various issues already found and fixed by the upstream maintainers since the inital release of Vim-7.2:

```
patch -Np1 -i ../vim-7.2-fixes-3.patch
```

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

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

Now prepare Vim for compilation:

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

The meaning of the configure options:

`--enable-multibyte`
: This switch enables support for editing files in multibyte character encodings. This is needed if using a locale with a multibyte character set. This switch is also helpful to be able to edit text files initially created in Linux distributions like Fedora Core that use UTF-8 as a default character set.

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

However, this test suite outputs a lot of binary data to the screen, which can cause issues with the settings of the current terminal. This can be resolved by redirecting the output to a log file.

Install the package:

```
make install
```

Many users are used to using **vi** instead of **vim**. To allow execution of **vim** when users habitually enter **vi**, create a symlink for both the binary and the man page in the provided languages:

```
ln -sv vim /usr/bin/vi
for L in  /usr/share/man/{,*/}man1/vim.1; do
    ln -sv vim.1 $(dirname $L)/vi.1
done
```

By default, Vim's documentation is installed in `/usr/share/vim`. The following symlink allows the documentation to be accessed via `/usr/share/doc/vim-7.2`, making it consistent with the location of documentation for other packages:

```
ln -sv ../vim/vim72/doc /usr/share/doc/vim-7.2
```

If an X Window System is going to be installed on the LFS system, it may be necessary to recompile Vim after installing X. Vim comes with a GUI version of the editor that requires X and some additional libraries to be installed. For more information on this process, refer to the Vim documentation and the Vim installation page in the BLFS book at *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.58.2. Configuring Vim

By default, **vim** runs in vi-incompatible mode. This may be new to users who have used other editors in the past. The “nocompatible” setting is included below to highlight the fact that a new behavior is being used. It also reminds those who would change to “compatible” mode that it should be the first setting in the configuration file. This is necessary because it changes other settings, and overrides must come after this setting. Create a default **vim** configuration file by running the following:

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

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

The *`set nocompatible`* setting makes **vim** behave in a more useful way (the default) than the vi-compatible manner. Remove the “no” to keep the old **vi** behavior. The *`set backspace=2`* setting allows backspacing over line breaks, autoindents, and the start of insert. The *`syntax on`* parameter enables vim's syntax highlighting. Finally, the *if* statement with the *`set background=dark`* setting corrects **vim**'s guess about the background color of some terminal emulators. This gives the highlighting a better color scheme for use on the black background of these programs.

Documentation for other available options can be obtained by running the following command:

```
vim -c ':options'
```

Note

By default, Vim only installs spell files for the English language. To install spell files for your preferred language, download the `*.spl` and optionally, the `*.sug` files for your language and character encoding from *ftp://ftp.vim.org/pub/vim/runtime/spell/* and save them to `/usr/share/vim/vim72/spell/`.

To use these spell files, some configuration in `/etc/vimrc` is needed, e.g.:

```
set spelllang=en,ru
set spell
```

For more information, see the appropriate README file located at the URL above.

6.58.3. Contents of Vim

Installed programs: 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, and xxd

Short Descriptions

ex	Starts **vim** in ex mode
rview	Is a restricted version of **view**; no shell commands can be started and **view** cannot be suspended
rvim	Is a restricted version of **vim**; no shell commands can be started and **vim** cannot be suspended
vi	Link to **vim**
view	Starts **vim** in read-only mode
vim	Is the editor
vimdiff	Edits two or three versions of a file with **vim** and show differences
vimtutor	Teaches the basic keys and commands of **vim**
xxd	Creates a hex dump of the given file; it can also do the reverse, so it can be used for binary patching

6.59. About Debugging Symbols

Most programs and libraries are, by default, compiled with debugging symbols included (with **gcc**'s *-g* option). This means that when debugging a program or library that was compiled with debugging information included, the debugger can provide not only memory addresses, but also the names of the routines and variables.

However, the inclusion of these debugging symbols enlarges a program or library significantly. The following is an example of the amount of space these symbols occupy:

- A **bash** binary with debugging symbols: 1200 KB
- A **bash** binary without debugging symbols: 480 KB
- Glibc and GCC files (`/lib` and `/usr/lib`) with debugging symbols: 87 MB
- Glibc and GCC files without debugging symbols: 16 MB

Sizes may vary depending on which compiler and C library were used, but when comparing programs with and without debugging symbols, the difference will usually be a factor between two and five.

Because most users will never use a debugger on their system software, a lot of disk space can be regained by removing these symbols. The next section shows how to strip all debugging symbols from the programs and libraries. Additional information on system optimization can be found at *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*.

6.60. Stripping Again

If the intended user is not a programmer and does not plan to do any debugging on the system software, the system size can be decreased by about 90 MB by removing the debugging symbols from binaries and libraries. This causes no inconvenience other than not being able to debug the software fully anymore.

Most people who use the command mentioned below do not experience any difficulties. However, it is easy to make a typo and render the new system unusable, so before running the **strip** command, it is a good idea to make a backup of the LFS system in its current state.

Before performing the stripping, take special care to ensure that none of the binaries that are about to be stripped are running. If unsure whether the user entered chroot with the command given in Section 6.4, “Entering the Chroot Environment,” first exit from chroot:

```
logout
```

Then reenter it with:

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

Now the binaries and libraries can be safely stripped:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
  -exec /tools/bin/strip --strip-debug '{}' ';'
```

A large number of files will be reported as having their file format not recognized. These warnings can be safely ignored. These warnings indicate that those files are scripts instead of binaries.

If disk space is very tight, the `--strip-all` option can be used on the binaries in `/{,usr/}{bin,sbin}` to gain several more megabytes. Do not use this option on libraries—they will be destroyed.

6.61. Cleaning Up

From now on, when reentering the chroot environment after exiting, use the following modified chroot command:

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

The reason for this is that the programs in `/tools` are no longer needed. Since they are no longer needed you can delete the `/tools` directory if so desired.

Note

Removing `/tools` will also remove the temporary copies of Tcl, Expect, and DejaGNU which were used for running the toolchain tests. If you need these programs later on, they will need to be recompiled and re-installed. The BLFS book has instructions for this (see *http://www.linuxfromscratch.org/blfs/*).

If the virtual kernel file systems have been unmounted, either manually or through a reboot, ensure that the virtual kernel file systems are mounted when reentering the chroot. This process was explained in Section 6.2.2, “Mounting and Populating /dev” and Section 6.2.3, “Mounting Virtual Kernel File Systems”.

Chapter 7. Setting Up System Bootscripts

7.1. Introduction

This chapter details how to install and configure the LFS-Bootscripts package. Most of these scripts will work without modification, but a few require additional configuration files because they deal with hardware-dependent information.

System-V style init scripts are employed in this book because they are widely used. For additional options, a hint detailing the BSD style init setup is available at *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. Searching the LFS mailing lists for “depinit” will also offer additional choices.

If using an alternative style of init scripts, skip this chapter and move on to Chapter 8.

7.2. LFS-Bootscripts-20081031

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

Approximate build time: less than 0.1 SBU
Required disk space: 464 KB

7.2.1. Installation of LFS-Bootscripts

Install the package:

```
make install
```

7.2.2. Contents of LFS-Bootscripts

Installed scripts: checkfs, cleanfs, console, consolelog, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysctl, sysklogd, template, udev, and udev_retry

Short Descriptions

checkfs Checks the integrity of the file systems before they are mounted (with the exception of journal and network based file systems)

cleanfs Removes files that should not be preserved between reboots, such as those in `/var/run/` and `/var/lock/`; it re-creates `/var/run/utmp` and removes the possibly present `/etc/nologin`, `/fastboot`, and `/forcefsck` files

console Loads the correct keymap table for the desired keyboard layout; it also sets the screen font

consolelog Sets the kernel log level to control messages reaching the console.

functions Contains common functions, such as error and status checking, that are used by several bootscripts

halt Halts the system

ifdown Assists the network script with stopping network devices

ifup Assists the network script with starting network devices

localnet Sets up the system's hostname and local loopback device

modules Loads kernel modules listed in `/etc/sysconfig/modules`, using arguments that are also given there

mountfs Mounts all file systems, except ones that are marked *noauto* or are network based

mountkernfs Mounts virtual kernel file systems, such as `proc`

network Sets up network interfaces, such as network cards, and sets up the default gateway (where applicable)

rc The master run-level control script; it is responsible for running all the other bootscripts one-by-one, in a sequence determined by the name of the symbolic links being processed

reboot Reboots the system

sendsignals Makes sure every process is terminated before the system reboots or halts

setclock Resets the kernel clock to local time in case the hardware clock is not set to UTC time

static	Provides the functionality needed to assign a static Internet Protocol (IP) address to a network interface
swap	Enables and disables swap files and partitions
sysctl	Loads system configuration values from `/etc/sysctl.conf`, if that file exists, into the running kernel
sysklogd	Starts and stops the system and kernel log daemons
template	A template to create custom bootscripts for other daemons
udev	Prepares the `/dev` directory and starts Udev
udev_retry	Retries failed udev uevents, and copies generated rules files from `/dev/.udev` to `/etc/udev/rules.d` if required

7.3. How Do These Bootscripts Work?

Linux uses a special booting facility named SysVinit that is based on a concept of *run-levels*. It can be quite different from one system to another, so it cannot be assumed that because things worked in one particular Linux distribution, they should work the same in LFS too. LFS has its own way of doing things, but it respects generally accepted standards.

SysVinit (which will be referred to as “init” from now on) works using a run-levels scheme. There are seven (numbered 0 to 6) run-levels (actually, there are more run-levels, but they are for special cases and are generally not used. See `init(8)` for more details), and each one of those corresponds to the actions the computer is supposed to perform when it starts up. The default run-level is 3. Here are the descriptions of the different run-levels as they are implemented:

```
0: halt the computer
1: single-user mode
2: multi-user mode without networking
3: multi-user mode with networking
4: reserved for customization, otherwise does the same as 3
5: same as 4, it is usually used for GUI login (like X's xdm or KDE's kdm)
6: reboot the computer
```

The command used to change run-levels is **init *<runlevel>***, where *<runlevel>* is the target run-level. For example, to reboot the computer, a user could issue the **init 6** command, which is an alias for the **reboot** command. Likewise, **init 0** is an alias for the **halt** command.

There are a number of directories under `/etc/rc.d` that look like `rc?.d` (where ? is the number of the run-level) and `rcsysinit.d`, all containing a number of symbolic links. Some begin with a *K*, the others begin with an *S*, and all of them have two numbers following the initial letter. The K means to stop (kill) a service and the S means to start a service. The numbers determine the order in which the scripts are run, from 00 to 99—the lower the number the earlier it gets executed. When **init** switches to another run-level, the appropriate services are either started or stopped, depending on the runlevel chosen.

The real scripts are in `/etc/rc.d/init.d`. They do the actual work, and the symlinks all point to them. Killing links and starting links point to the same script in `/etc/rc.d/init.d`. This is because the scripts can be called with different parameters like *start*, *stop*, *restart*, *reload*, and *status*. When a K link is encountered, the appropriate script is run with the *stop* argument. When an S link is encountered, the appropriate script is run with the *start* argument.

There is one exception to this explanation. Links that start with an *S* in the `rc0.d` and `rc6.d` directories will not cause anything to be started. They will be called with the parameter *stop* to stop something. The logic behind this is that when a user is going to reboot or halt the system, nothing needs to be started. The system only needs to be stopped.

These are descriptions of what the arguments make the scripts do:

start
: The service is started.

stop
: The service is stopped.

restart
: The service is stopped and then started again.

reload

The configuration of the service is updated. This is used after the configuration file of a service was modified, when the service does not need to be restarted.

status

Tells if the service is running and with which PIDs.

Feel free to modify the way the boot process works (after all, it is your own LFS system). The files given here are an example of how it can be done.

7.4. Device and Module Handling on an LFS System

In Chapter 6, we installed the Udev package. Before we go into the details regarding how this works, a brief history of previous methods of handling devices is in order.

Linux systems in general traditionally use a static device creation method, whereby a great many device nodes are created under `/dev` (sometimes literally thousands of nodes), regardless of whether the corresponding hardware devices actually exist. This is typically done via a **MAKEDEV** script, which contains a number of calls to the **mknod** program with the relevant major and minor device numbers for every possible device that might exist in the world.

Using the Udev method, only those devices which are detected by the kernel get device nodes created for them. Because these device nodes will be created each time the system boots, they will be stored on a `tmpfs` file system (a virtual file system that resides entirely in system memory). Device nodes do not require much space, so the memory that is used is negligible.

7.4.1. History

In February 2000, a new filesystem called `devfs` was merged into the 2.3.46 kernel and was made available during the 2.4 series of stable kernels. Although it was present in the kernel source itself, this method of creating devices dynamically never received overwhelming support from the core kernel developers.

The main problem with the approach adopted by `devfs` was the way it handled device detection, creation, and naming. The latter issue, that of device node naming, was perhaps the most critical. It is generally accepted that if device names are allowed to be configurable, then the device naming policy should be up to a system administrator, not imposed on them by any particular developer(s). The `devfs` file system also suffers from race conditions that are inherent in its design and cannot be fixed without a substantial revision to the kernel. It was marked as deprecated for a long period – due to a lack of maintenance – and was finally removed from the kernel in June, 2006.

With the development of the unstable 2.5 kernel tree, later released as the 2.6 series of stable kernels, a new virtual filesystem called `sysfs` came to be. The job of `sysfs` is to export a view of the system's hardware configuration to userspace processes. With this userspace-visible representation, the possibility of seeing a userspace replacement for `devfs` became much more realistic.

7.4.2. Udev Implementation

7.4.2.1. Sysfs

The `sysfs` filesystem was mentioned briefly above. One may wonder how `sysfs` knows about the devices present on a system and what device numbers should be used for them. Drivers that have been compiled into the kernel directly register their objects with `sysfs` as they are detected by the kernel. For drivers compiled as modules, this registration will happen when the module is loaded. Once the `sysfs` filesystem is mounted (on `/sys`), data which the built-in drivers registered with `sysfs` are available to userspace processes and to **udevd** for device node creation.

7.4.2.2. Udev Bootscript

The **S10udev** initscript takes care of creating device nodes when Linux is booted. The script unsets the uevent handler from the default of **/sbin/hotplug**. This is done because the kernel no longer needs to call out to an external binary. Instead **udevd** will listen on a netlink socket for uevents that the kernel raises. Next, the bootscript copies any static device nodes that exist in `/lib/udev/devices` to `/dev`. This is necessary because some devices, directories, and symlinks are needed before the dynamic device handling processes are available during the early stages of booting a system, or are required by **udevd** itself. Creating static device nodes in `/lib/udev/devices` also provides an easy workaround for devices that are not supported by the dynamic device handling infrastructure. The bootscript then starts the Udev daemon, **udevd**, which will act on any uevents it receives. Finally, the bootscript forces the kernel to replay uevents for any devices that have already been registered and then waits for **udevd** to handle them.

7.4.2.3. Device Node Creation

To obtain the right major and minor number for a device, Udev relies on the information provided by `sysfs` in `/sys`. For example, `/sys/class/tty/vcs/dev` contains the string “7:0”. This string is used by **udevd** to create a device node with major number *7* and minor *0*. The names and permissions of the nodes created under the `/dev` directory are determined by rules specified in the files within the `/etc/udev/rules.d/` directory. These are numbered in a similar fashion to the LFS-Bootscripts package. If **udevd** can't find a rule for the device it is creating, it will default permissions to *660* and ownership to *root:root*. Documentation on the syntax of the Udev rules configuration files are available in `/usr/share/doc/udev-130/index.html`

7.4.2.4. Module Loading

Device drivers compiled as modules may have aliases built into them. Aliases are visible in the output of the **modinfo** program and are usually related to the bus-specific identifiers of devices supported by a module. For example, the *snd-fm801* driver supports PCI devices with vendor ID 0x1319 and device ID 0x0801, and has an alias of “pci:v00001319d00000801sv*sd*bc04sc01i*”. For most devices, the bus driver exports the alias of the driver that would handle the device via `sysfs`. E.g., the `/sys/bus/pci/devices/0000:00:0d.0/modalias` file might contain the string “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. The default rules provided with Udev will cause **udevd** to call out to **/sbin/modprobe** with the contents of the `MODALIAS` uevent environment variable (which should be the same as the contents of the `modalias` file in sysfs), thus loading all modules whose aliases match this string after wildcard expansion.

In this example, this means that, in addition to *snd-fm801*, the obsolete (and unwanted) *forte* driver will be loaded if it is available. See below for ways in which the loading of unwanted drivers can be prevented.

The kernel itself is also able to load modules for network protocols, filesystems and NLS support on demand.

7.4.2.5. Handling Hotpluggable/Dynamic Devices

When you plug in a device, such as a Universal Serial Bus (USB) MP3 player, the kernel recognizes that the device is now connected and generates a uevent. This uevent is then handled by **udevd** as described above.

7.4.3. Problems with Loading Modules and Creating Devices

There are a few possible problems when it comes to automatically creating device nodes.

7.4.3.1. A kernel module is not loaded automatically

Udev will only load a module if it has a bus-specific alias and the bus driver properly exports the necessary aliases to `sysfs`. In other cases, one should arrange module loading by other means. With Linux-2.6.27.4, Udev is known to load properly-written drivers for INPUT, IDE, PCI, USB, SCSI, SERIO and FireWire devices.

To determine if the device driver you require has the necessary support for Udev, run **modinfo** with the module name as the argument. Now try locating the device directory under `/sys/bus` and check whether there is a `modalias` file there.

If the `modalias` file exists in `sysfs`, the driver supports the device and can talk to it directly, but doesn't have the alias, it is a bug in the driver. Load the driver without the help from Udev and expect the issue to be fixed later.

If there is no `modalias` file in the relevant directory under `/sys/bus`, this means that the kernel developers have not yet added modalias support to this bus type. With Linux-2.6.27.4, this is the case with ISA busses. Expect this issue to be fixed in later kernel versions.

Udev is not intended to load "wrapper" drivers such as *snd-pcm-oss* and non-hardware drivers such as *loop* at all.

7.4.3.2. A kernel module is not loaded automatically, and Udev is not intended to load it

If the "wrapper" module only enhances the functionality provided by some other module (e.g., *snd-pcm-oss* enhances the functionality of *snd-pcm* by making the sound cards available to OSS applications), configure **modprobe** to load the wrapper after Udev loads the wrapped module. To do this, add an "install" line in `/etc/modprobe.conf`. For example:

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

If the module in question is not a wrapper and is useful by itself, configure the **S05modules** bootscript to load this module on system boot. To do this, add the module name to the `/etc/sysconfig/modules` file on a separate line. This works for wrapper modules too, but is suboptimal in that case.

7.4.3.3. Udev loads some unwanted module

Either don't build the module, or blacklist it in `/etc/modprobe.conf` file as done with the *forte* module in the example below:

```
blacklist forte
```

Blacklisted modules can still be loaded manually with the explicit **modprobe** command.

7.4.3.4. Udev creates a device incorrectly, or makes a wrong symlink

This usually happens if a rule unexpectedly matches a device. For example, a poorly-writen rule can match both a SCSI disk (as desired) and the corresponding SCSI generic device (incorrectly) by vendor. Find the offending rule and make it more specific, with the help of the **udevadm info** command.

7.4.3.5. Udev rule works unreliably

This may be another manifestation of the previous problem. If not, and your rule uses `sysfs` attributes, it may be a kernel timing issue, to be fixed in later kernels. For now, you can work around it by creating a rule that waits for the used `sysfs` attribute and appending it to the `/etc/udev/rules.d/10-wait_for_sysfs.rules` file (create this file if it does not exist). Please notify the LFS Development list if you do so and it helps.

7.4.3.6. Udev does not create a device

Further text assumes that the driver is built statically into the kernel or already loaded as a module, and that you have already checked that Udev doesn't create a misnamed device.

Udev has no information needed to create a device node if a kernel driver does not export its data to `sysfs`. This is most common with third party drivers from outside the kernel tree. Create a static device node in `/lib/udev/devices` with the appropriate major/minor numbers (see the file `devices.txt` inside the kernel documentation or the documentation provided by the third party driver vendor). The static device node will be copied to `/dev` by the **S10udev** bootscript.

7.4.3.7. Device naming order changes randomly after rebooting

This is due to the fact that Udev, by design, handles uevents and loads modules in parallel, and thus in an unpredictable order. This will never be “fixed”. You should not rely upon the kernel device names being stable. Instead, create your own rules that make symlinks with stable names based on some stable attributes of the device, such as a serial number or the output of various *_id utilities installed by Udev. See Section 7.12, “Creating Custom Symlinks to Devices” and Section 7.13, “Configuring the network Script” for examples.

7.4.4. Useful Reading

Additional helpful documentation is available at the following sites:

- A Userspace Implementation of `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- udev FAQ *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ*
- The `sysfs` Filesystem *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

7.5. Configuring the setclock Script

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

If you cannot remember whether or not the hardware clock is set to UTC, find out by running the **`hwclock --localtime --show`** command. This will display what the current time is according to the hardware clock. If this time matches whatever your watch says, then the hardware clock is set to local time. If the output from **hwclock** is not local time, chances are it is set to UTC time. Verify this by adding or subtracting the proper amount of hours for the timezone to the time shown by **hwclock**. For example, if you are currently in the MST timezone, which is also known as GMT -0700, add seven hours to the local time.

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

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

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

UTC=1

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

A good hint explaining how to deal with time on LFS is available at *http://www.linuxfromscratch.org/hints/downloads/files/time.txt*. It explains issues such as time zones, UTC, and the `TZ` environment variable.

7.6. Configuring the Linux Console

This section discusses how to configure the **console** and **consolelog** bootscripts that set up the keyboard map, console font and console kernel log level. If non-ASCII characters (e.g., the copyright sign, the British pound sign and Euro symbol) will not be used and the keyboard is a U.S. one, much of this section can be skipped. Without the configuration file, the **console** bootscript will do nothing.

The **console** and **consolelog** script reads the `/etc/sysconfig/console` file for configuration information. Decide which keymap and screen font will be used. Various language-specific HOWTOs can also help with this, see *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. If still in doubt, look in the `/lib/kbd` directory for valid keymaps and screen fonts. Read `loadkeys(1)` and `setfont(8)` manual pages to determine the correct arguments for these programs.

The `/etc/sysconfig/console` file should contain lines of the form: VARIABLE="value". The following variables are recognized:

LOGLEVEL

This variable specifies the log level for kernel messages sent to the console as set by **dmesg**. Valid levels are from "1" (no messages) to "8". The default level is "7".

KEYMAP

This variable specifies the arguments for the **loadkeys** program, typically, the name of keymap to load, e.g., “es”. If this variable is not set, the bootscript will not run the **loadkeys** program, and the default kernel keymap will be used.

KEYMAP_CORRECTIONS

This (rarely used) variable specifies the arguments for the second call to the **loadkeys** program. This is useful if the stock keymap is not completely satisfactory and a small adjustment has to be made. E.g., to include the Euro sign into a keymap that normally doesn't have it, set this variable to “euro2”.

FONT

This variable specifies the arguments for the **setfont** program. Typically, this includes the font name, “-m”, and the name of the application character map to load. E.g., in order to load the “lat1-16” font together with the “8859-1” application character map (as it is appropriate in the USA), set this variable to “lat1-16 -m 8859-1”. In UTF-8 mode, the kernel uses the application character map for conversion of composed 8-bit key codes in the keymap to UTF-8, and thus the argument of the "-m" parameter should be set to the encoding of the composed key codes in the keymap.

UNICODE

Set this variable to “1”, “yes” or “true” in order to put the console into UTF-8 mode. This is useful in UTF-8 based locales and harmful otherwise.

LEGACY_CHARSET

For many keyboard layouts, there is no stock Unicode keymap in the Kbd package. The **console** bootscript will convert an available keymap to UTF-8 on the fly if this variable is set to the encoding of the available non-UTF-8 keymap.

Some examples:

- For a non-Unicode setup, only the KEYMAP and FONT variables are generally needed. E.g., for a Polish setup, one would use:

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

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- As mentioned above, it is sometimes necessary to adjust a stock keymap slightly. The following example adds the Euro symbol to the German keymap:

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

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- The following is a Unicode-enabled example for Bulgarian, where a stock UTF-8 keymap exists:

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

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- Due to the use of a 512-glyph LatArCyrHeb-16 font in the previous example, bright colors are no longer available on the Linux console unless a framebuffer is used. If one wants to have bright colors without framebuffer and can live without characters not belonging to his language, it is still possible to use a language-specific 256-glyph font, as illustrated below:

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

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- The following example illustrates keymap autoconversion from ISO-8859-15 to UTF-8 and enabling dead keys in Unicode mode:

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

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- Some keymaps have dead keys (i.e., keys that don't produce a character by themselves, but put an accent on the character produced by the next key) or define composition rules (such as: “press Ctrl+. A E to get Æ” in the default keymap). Linux-2.6.27.4 interprets dead keys and composition rules in the keymap correctly only when the source characters to be composed together are not multibyte. This deficiency doesn't affect keymaps for European languages, because there accents are added to unaccented ASCII characters, or two ASCII characters are composed together. However, in UTF-8 mode it is a problem, e.g., for the Greek language, where one sometimes needs to put an accent on the letter “alpha”. The solution is either to avoid the use of UTF-8, or to install the X window system that doesn't have this limitation in its input handling.
- For Chinese, Japanese, Korean and some other languages, the Linux console cannot be configured to display the needed characters. Users who need such languages should install the X Window System, fonts that cover the necessary character ranges, and the proper input method (e.g., SCIM, it supports a wide variety of languages).

Note

The `/etc/sysconfig/console` file only controls the Linux text console localization. It has nothing to do with setting the proper keyboard layout and terminal fonts in the X Window System, with ssh sessions or with a serial console. In such situations, limitations mentioned in the last two list items above do not apply.

7.7. Configuring the sysklogd Script

The `sysklogd` script invokes the **syslogd** program with the *`-m 0`* option. This option turns off the periodic timestamp mark that **syslogd** writes to the log files every 20 minutes by default. If you want to turn on this periodic timestamp mark, edit the `sysklogd` script and make the changes accordingly. See **`man syslogd`** for more information.

7.8. Creating the /etc/inputrc File

The `inputrc` file handles keyboard mapping for specific situations. This file is the startup file used by Readline — the input-related library — used by Bash and most other shells.

Most people do not need user-specific keyboard mappings so the command below creates a global `/etc/inputrc` used by everyone who logs in. If you later decide you need to override the defaults on a per-user basis, you can create a `.inputrc` file in the user's home directory with the modified mappings.

For more information on how to edit the `inputrc` file, see **info bash** under the *Readline Init File* section. **info readline** is also a good source of information.

Below is a generic global `inputrc` along with comments to explain what the various options do. Note that comments cannot be on the same line as commands. Create the file using the following command:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. The Bash Shell Startup Files

The shell program **/bin/bash** (hereafter referred to as “the shell”) uses a collection of startup files to help create an environment to run in. Each file has a specific use and may affect login and interactive environments differently. The files in the `/etc` directory provide global settings. If an equivalent file exists in the home directory, it may override the global settings.

An interactive login shell is started after a successful login, using **/bin/login**, by reading the `/etc/passwd` file. An interactive non-login shell is started at the command-line (e.g., `[prompt]$`**/bin/bash**). A non-interactive shell is usually present when a shell script is running. It is non-interactive because it is processing a script and not waiting for user input between commands.

For more information, see **info bash** under the *Bash Startup Files and Interactive Shells* section.

The files `/etc/profile` and `~/.bash_profile` are read when the shell is invoked as an interactive login shell.

The base `/etc/profile` below sets some environment variables necessary for native language support. Setting them properly results in:

- The output of programs translated into the native language
- Correct classification of characters into letters, digits and other classes. This is necessary for **bash** to properly accept non-ASCII characters in command lines in non-English locales
- The correct alphabetical sorting order for the country
- Appropriate default paper size
- Correct formatting of monetary, time, and date values

Replace `<ll>` below with the two-letter code for the desired language (e.g., “en”) and `<CC>` with the two-letter code for the appropriate country (e.g., “GB”). `<charmap>` should be replaced with the canonical charmap for your chosen locale. Optional modifiers such as “@euro” may also be present.

The list of all locales supported by Glibc can be obtained by running the following command:

```
locale -a
```

Charmaps can have a number of aliases, e.g., “ISO-8859-1” is also referred to as “iso8859-1” and “iso88591”. Some applications cannot handle the various synonyms correctly (e.g., require that “UTF-8” is written as “UTF-8”, not “utf8”), so it is safest in most cases to choose the canonical name for a particular locale. To determine the canonical name, run the following command, where `<locale name>` is the output given by **locale -a** for your preferred locale (“en_GB.iso88591” in our example).

```
LC_ALL=<locale name> locale charmap
```

For the “en_GB.iso88591” locale, the above command will print:

```
ISO-8859-1
```

This results in a final locale setting of “en_GB.ISO-8859-1”. It is important that the locale found using the heuristic above is tested prior to it being added to the Bash startup files:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

The above commands should print the language name, the character encoding used by the locale, the local currency, and the prefix to dial before the telephone number in order to get into the country. If any of the commands above fail with a message similar to the one shown below, this means that your locale was either not installed in Chapter 6 or is not supported by the default installation of Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

If this happens, you should either install the desired locale using the **localedef** command, or consider choosing a different locale. Further instructions assume that there are no such error messages from Glibc.

Some packages beyond LFS may also lack support for your chosen locale. One example is the X library (part of the X Window System), which outputs the following error message if the locale does not exactly match one of the character map names in its internal files:

```
Warning: locale not supported by Xlib, locale set to C
```

In several cases Xlib expects that the character map will be listed in uppercase notation with canonical dashes. For instance, "ISO-8859-1" rather than "iso88591". It is also possible to find an appropriate specification by removing the charmap part of the locale specification. This can be checked by running the **locale charmap** command in both locales. For example, one would have to change "de_DE.ISO-8859-15@euro" to "de_DE@euro" in order to get this locale recognized by Xlib.

Other packages can also function incorrectly (but may not necessarily display any error messages) if the locale name does not meet their expectations. In those cases, investigating how other Linux distributions support your locale might provide some useful information.

Once the proper locale settings have been determined, create the `/etc/profile` file:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

The “C” (default) and “en_US” (the recommended one for United States English users) locales are different. “C” uses the US-ASCII 7-bit character set, and treats bytes with the high bit set as invalid characters. That's why, e.g., the **ls** command substitutes them with question marks in that locale. Also, an attempt to send mail with such characters from Mutt or Pine results in non-RFC-conforming messages being sent (the charset in the outgoing mail is indicated as “unknown 8-bit”). So you can use the “C” locale only if you are sure that you will never need 8-bit characters.

UTF-8 based locales are not supported well by many programs. E.g., the **watch** program displays only ASCII characters in UTF-8 locales and has no such restriction in traditional 8-bit locales like en_US. Work is in progress to document and, if possible, fix such problems, see *http://www.linuxfromscratch.org/blfs/view/svn/introduction/locale-issues.html*.

7.10. Configuring the localnet Script

Part of the job of the **localnet** script is setting the system's hostname. This needs to be configured in the `/etc/sysconfig/network` file.

Create the `/etc/sysconfig/network` file and enter a hostname by running:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

`<lfs>` needs to be replaced with the name given to the computer. Do not enter the Fully Qualified Domain Name (FQDN) here. That information will be put in the `/etc/hosts` file in the next section.

7.11. Customizing the /etc/hosts File

If a network card is to be configured, decide on the IP address, fully-qualified domain name (FQDN), and possible aliases for use in the `/etc/hosts` file. The syntax is:

```
IP_address myhost.example.org aliases
```

Unless the computer is to be visible to the Internet (i.e., there is a registered domain and a valid block of assigned IP addresses—most users do not have this), make sure that the IP address is in the private network IP address range. Valid ranges are:

```
Private Network Address Range      Normal Prefix
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x can be any number in the range 16-31. y can be any number in the range 0-255.

A valid private IP address could be 192.168.1.1. A valid FQDN for this IP could be lfs.example.org.

Even if not using a network card, a valid FQDN is still required. This is necessary for certain programs to operate correctly.

Create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# End /etc/hosts (network card version)
EOF
```

The *`<192.168.1.1>`* and *`<HOSTNAME.example.org>`* values need to be changed for specific users or requirements (if assigned an IP address by a network/system administrator and the machine will be connected to an existing network). The optional alias name(s) can be omitted.

If a network card is not going to be configured, create the `/etc/hosts` file by running:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

7.12. Creating Custom Symlinks to Devices

7.12.1. CD-ROM symlinks

Some software that you may want to install later (e.g., various media players) expect the `/dev/cdrom` and `/dev/dvd` symlinks to exist, and to point to a CD-ROM or DVD-ROM device. Also, it may be convenient to put references to those symlinks into `/etc/fstab`. Udev comes with a script that will generate rules files to create these symlinks for you, depending on the capabilities of each device, but you need to decide which of two modes of operation you wish to have the script use.

First, the script can operate in “by-path” mode (used by default for USB and FireWire devices), where the rules it creates depend on the physical path to the CD or DVD device. Second, it can operate in “by-id” mode (default for IDE and SCSI devices), where the rules it creates depend on identification strings stored in the CD or DVD device itself. The path is determined by Udev's **path_id** script, and the identification strings are read from the hardware by its **ata_id** or **scsi_id** programs, depending on which type of device you have.

There are advantages to each approach; the correct approach to use will depend on what kinds of device changes may happen. If you expect the physical path to the device (that is, the ports and/or slots that it plugs into) to change, for example because you plan on moving the drive to a different IDE port or a different USB connector, then you should use the “by-id” mode. On the other hand, if you expect the device's identification to change, for example because it may die, and you would replace it with a different device with the same capabilities and which is plugged into the same connectors, then you should use the “by-path” mode.

If either type of change is possible with your drive, then choose a mode based on the type of change you expect to happen more often.

Important

External devices (for example, a USB-connected CD drive) should not use by-path persistence, because each time the device is plugged into a new external port, its physical path will change. All externally-connected devices will have this problem if you write Udev rules to recognize them by their physical path; the problem is not limited to CD and DVD drives.

If you wish to see the values that the Udev scripts will use, then for the appropriate CD-ROM device, find the corresponding directory under `/sys` (e.g., this can be `/sys/block/hdd`) and run a command similar to the following:

```
udevadm test /sys/block/hdd
```

Look at the lines containing the output of various *_id programs. The “by-id” mode will use the ID_SERIAL value if it exists and is not empty, otherwise it will use a combination of ID_MODEL and ID_REVISION. The “by-path” mode will use the ID_PATH value.

If the default mode is not suitable for your situation, then the following modification can be made to the `/etc/udev/rules.d/75-cd-aliases-generator.rules` file, as follows (where *`mode`* is one of “by-id” or “by-path”):

```
sed -i -e 's/write_cd_rules/& mode/' \
    /etc/udev/rules.d/75-cd-aliases-generator.rules
```

Note that it is not necessary to create the rules files or symlinks at this time, because you have bind-mounted the host's `/dev` directory into the LFS system, and we assume the symlinks exist on the host. The rules and symlinks will be created the first time you boot your LFS system.

However, if you have multiple CD-ROM devices, then the symlinks generated at that time may point to different devices than they point to on your host, because devices are not discovered in a predictable order. The assignments created when you first boot the LFS system will be stable, so this is only an issue if you need the symlinks on both systems to point to the same device. If you need that, then inspect (and possibly edit) the generated `/etc/udev/rules.d/70-persistent-cd.rules` file after booting, to make sure the assigned symlinks match what you need.

7.12.2. Dealing with duplicate devices

As explained in Section 7.4, “Device and Module Handling on an LFS System”, the order in which devices with the same function appear in `/dev` is essentially random. E.g., if you have a USB web camera and a TV tuner, sometimes `/dev/video0` refers to the camera and `/dev/video1` refers to the tuner, and sometimes after a reboot the order changes to the opposite one. For all classes of hardware except sound cards and network cards, this is fixable by creating udev rules for custom persistent symlinks. The case of network cards is covered separately in Section 7.13, “Configuring the network Script”, and sound card configuration can be found in *BLFS*.

For each of your devices that is likely to have this problem (even if the problem doesn't exist in your current Linux distribution), find the corresponding directory under `/sys/class` or `/sys/block`. For video devices, this may be `/sys/class/video4linux/videoX`. Figure out the attributes that identify the device uniquely (usually, vendor and product IDs and/or serial numbers work):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Then write rules that create the symlinks, e.g.:

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Persistent symlinks for webcam and tuner
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f", ATTRS{vendor}=="0x109e", \
    SYMLINK+="tvtuner"

EOF
```

The result is that `/dev/video0` and `/dev/video1` devices still refer randomly to the tuner and the web camera (and thus should never be used directly), but there are symlinks `/dev/tvtuner` and `/dev/webcam` that always point to the correct device.

More information on writing Udev rules can be found in `/usr/share/doc/udev-130/index.html`.

7.13. Configuring the network Script

This section only applies if a network card is to be configured.

If a network card will not be used, there is likely no need to create any configuration files relating to network cards. If that is the case, remove the `network` symlinks from all run-level directories (`/etc/rc.d/rc*.d`).

7.13.1. Creating stable names for network interfaces

With Udev and modular network drivers, the network interface numbering is not persistent across reboots by default, because the drivers are loaded in parallel and, thus, in random order. For example, on a computer having two network cards made by Intel and Realtek, the network card manufactured by Intel may become `eth0` and the Realtek card becomes `eth1`. In some cases, after a reboot the cards get renumbered the other way around. To avoid this, Udev comes with a script and some rules to assign stable names to network cards based on their MAC address.

Pre-generate the rules to ensure the same names get assigned to the same devices at every boot, including the first:

```
for NIC in /sys/class/net/* ; do
    INTERFACE=${NIC##*/} udevadm test --action=add --subsystem=net $NIC
done
```

Now, inspect the `/etc/udev/rules.d/70-persistent-net.rules` file, to find out which name was assigned to which network device:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

The file begins with a comment block followed by two lines for each NIC. The first line for each NIC is a commented description showing its hardware IDs (e.g. its PCI vendor and device IDs, if it's a PCI card), along with its driver in parentheses, if the driver can be found. Neither the hardware ID nor the driver is used to determine which name to give an interface; this information is only for reference. The second line is the Udev rule that matches this NIC and actually assigns it a name.

All Udev rules are made up of several keys, separated by commas and optional whitespace. This rule's keys and an explanation of each of them are as follows:

- `SUBSYSTEM=="net"` - This tells Udev to ignore devices that are not network cards.
- `ACTION=="add"` - This tells Udev to ignore this rule for a uevent that isn't an add ("remove" and "change" uevents also happen, but don't need to rename network interfaces).
- `DRIVERS=="?*"` - This exists so that Udev will ignore VLAN or bridge sub-interfaces (because these sub-interfaces do not have drivers). These sub-interfaces are skipped because the name that would be assigned would collide with their parent devices.
- `ATTR{address}` - The value of this key is the NIC's MAC address.
- `ATTR{type}=="1"` - This ensures the rule only matches the primary interface in the case of certain wireless drivers, which create multiple virtual interfaces. The secondary interfaces are skipped for the same reason that VLAN and bridge sub-interfaces are skipped: there would be a name collision otherwise.
- `KERNEL=="eth*"` - This key was added to the Udev rule generator to handle machines that have multiple network interfaces, all with the same MAC address (the PS3 is one such machine). If the independent interfaces have different basenames, this key will allow Udev to tell them apart. This is generally not necessary for most Linux From Scratch users, but does not hurt.
- `NAME` - The value of this key is the name that Udev will assign to this interface.

The value of `NAME` is the important part. Make sure you know which name has been assigned to each of your network cards before proceeding, and be sure to use that `NAME` value when creating your configuration files below.

7.13.2. Creating Network Interface Configuration Files

Which interfaces are brought up and down by the network script depends on the files and directories in the `/etc/sysconfig/network-devices` hierarchy. This directory should contain a sub-directory for each interface to be configured, such as `ifconfig.xyz`, where “xyz” is a network interface name. Inside this directory would be files defining the attributes to this interface, such as its IP address(es), subnet masks, and so forth.

The following command creates a sample `ipv4` file for the *eth0* device:

```
cd /etc/sysconfig/network-devices
mkdir -v ifconfig.eth0
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

The values of these variables must be changed in every file to match the proper setup. If the `ONBOOT` variable is set to “yes” the network script will bring up the Network Interface Card (NIC) during booting of the system. If set to anything but “yes” the NIC will be ignored by the network script and not be brought up.

The `SERVICE` variable defines the method used for obtaining the IP address. The LFS-Bootscripts package has a modular IP assignment format, and creating additional files in the `/etc/sysconfig/network-devices/services` directory allows other IP assignment methods. This is commonly used for Dynamic Host Configuration Protocol (DHCP), which is addressed in the BLFS book.

The `GATEWAY` variable should contain the default gateway IP address, if one is present. If not, then comment out the variable entirely.

The `PREFIX` variable needs to contain the number of bits used in the subnet. Each octet in an IP address is 8 bits. If the subnet's netmask is 255.255.255.0, then it is using the first three octets (24 bits) to specify the network number. If the netmask is 255.255.255.240, it would be using the first 28 bits. Prefixes longer than 24 bits are commonly used by DSL and cable-based Internet Service Providers (ISPs). In this example (PREFIX=24), the netmask is 255.255.255.0. Adjust the `PREFIX` variable according to your specific subnet.

7.13.3. Creating the /etc/resolv.conf File

If the system is going to be connected to the Internet, it will need some means of Domain Name Service (DNS) name resolution to resolve Internet domain names to IP addresses, and vice versa. This is best achieved by placing the IP address of the DNS server, available from the ISP or network administrator, into `/etc/resolv.conf`. Create the file by running the following:

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

domain <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>

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

Replace `<IP address of the nameserver>` with the IP address of the DNS most appropriate for the setup. There will often be more than one entry (requirements demand secondary servers for fallback capability). If you only need or want one DNS server, remove the second *nameserver* line from the file. The IP address may also be a router on the local network.

Chapter 8. Making the LFS System Bootable

8.1. Introduction

It is time to make the LFS system bootable. This chapter discusses creating an `fstab` file, building a kernel for the new LFS system, and installing the GRUB boot loader so that the LFS system can be selected for booting at startup.

8.2. Creating the /etc/fstab File

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

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

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

/dev/<xxx>     /            <fff>  defaults        1     1
/dev/<yyy>     swap         swap   pri=1           0     0
proc           /proc        proc   defaults        0     0
sysfs          /sys         sysfs  defaults        0     0
devpts         /dev/pts     devpts gid=4,mode=620  0     0
tmpfs          /dev/shm     tmpfs  defaults        0     0
# End /etc/fstab
EOF
```

Replace *<xxx>*, *<yyy>*, and *<fff>* with the values appropriate for the system, for example, `hda2`, `hda5`, and `ext3`. For details on the six fields in this file, see **man 5 fstab**.

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

Filesystems with MS-DOS or Windows origin (i.e.: vfat, ntfs, smbfs, cifs, iso9660, udf) need the “iocharset” mount option in order for non-ASCII characters in file names to be interpreted properly. The value of this option should be the same as the character set of your locale, adjusted in such a way that the kernel understands it. This works if the relevant character set definition (found under File systems -> Native Language Support) has been compiled into the kernel or built as a module. The “codepage” option is also needed for vfat and smbfs filesystems. It should be set to the codepage number used under MS-DOS in your country. E.g., in order to mount USB flash drives, a ru_RU.KOI8-R user would need the following in the options portion of its mount line in `/etc/fstab`:

```
noauto,user,quiet,showexec,iocharset=koi8r,codepage=866
```

The corresponding options fragment for ru_RU.UTF-8 users is:

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

> **Note**
>
> In the latter case, the kernel emits the following message:
>
> ```
> FAT: utf8 is not a recommended IO charset for FAT filesystems,
>     filesystem will be case sensitive!
> ```
>
> This negative recommendation should be ignored, since all other values of the “iocharset” option result in wrong display of filenames in UTF-8 locales.

It is also possible to specify default codepage and iocharset values for some filesystems during kernel configuration. The relevant parameters are named “Default NLS Option” (`CONFIG_NLS_DEFAULT)`, “Default Remote NLS Option” (`CONFIG_SMB_NLS_DEFAULT`), “Default codepage for FAT” (`CONFIG_FAT_DEFAULT_CODEPAGE`), and “Default iocharset for FAT” (`CONFIG_FAT_DEFAULT_IOCHARSET`). There is no way to specify these settings for the ntfs filesystem at kernel compilation time.

8.3. Linux-2.6.27.4

The Linux package contains the Linux kernel.

Approximate build time: 1.5 - 5.0 SBU
Required disk space: 350 - 500 MB

8.3.1. Installation of the kernel

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

Prepare for compilation by running the following command:

```
make mrproper
```

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

Configure the kernel via a menu-driven interface. BLFS has some information regarding particular kernel configuration requirements of packages outside of LFS at *http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index*:

```
make LANG=<host_LANG_value> LC_ALL= menuconfig
```

The meaning of the make parameters:

`LANG=<host_LANG_value> LC_ALL=`
: This establishes the locale setting to the one used on the host. This is needed for a proper menuconfig ncurses interface line drawing on UTF-8 linux text console.
Be sure to replace *`<host_LANG_value>`* by the value of the `$LANG` variable from your host. If not set, you could use instead the host's value of `$LC_ALL` or `$LC_CTYPE`.

Alternatively, **make oldconfig** may be more appropriate in some situations. See the `README` file for more information.

If desired, skip kernel configuration by copying the kernel config file, `.config`, from the host system (assuming it is available) to the unpacked `linux-2.6.27.4` directory. However, we do not recommend this option. It is often better to explore all the configuration menus and create the kernel configuration from scratch.

Compile the kernel image and modules:

```
make
```

If using kernel modules, an `/etc/modprobe.conf` file may be needed. Information pertaining to modules and kernel configuration is located in Section 7.4, “Device and Module Handling on an LFS System” and in the kernel documentation in the `linux-2.6.27.4/Documentation` directory. Also, `modprobe.conf(5)` may be of interest.

Install the modules, if the kernel configuration uses them:

```
make modules_install
```

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

The path to the kernel image may vary depending on the platform being used. The following command assumes an x86 architecture:

```
cp -v arch/x86/boot/bzImage /boot/lfskernel-2.6.27.4
```

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

```
cp -v System.map /boot/System.map-2.6.27.4
```

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

```
cp -v .config /boot/config-2.6.27.4
```

Install the documentation for the Linux kernel:

```
install -d /usr/share/doc/linux-2.6.27.4
cp -r Documentation/* /usr/share/doc/linux-2.6.27.4
```

It is important to note that the files in the kernel source directory are not owned by *root*. Whenever a package is unpacked as user *root* (like we did inside chroot), the files have the user and group IDs of whatever they were on the packager's computer. This is usually not a problem for any other package to be installed because the source tree is removed after the installation. However, the Linux source tree is often retained for a long time. Because of this, there is a chance that whatever user ID the packager used will be assigned to somebody on the machine. That person would then have write access to the kernel source.

If the kernel source tree is going to be retained, run **chown -R 0:0** on the `linux-2.6.27.4` directory to ensure all files are owned by user *root*.

Warning

Some kernel documentation recommends creating a symlink from `/usr/src/linux` pointing to the kernel source directory. This is specific to kernels prior to the 2.6 series and *must not* be created on an LFS system as it can cause problems for packages you may wish to build once your base LFS system is complete.

Warning

The headers in the system's `include` directory should *always* be the ones against which Glibc was compiled, that is, the sanitised headers from this Linux kernel tarball. Therefore, they should *never* be replaced by either the raw kernel headers or any other kernel sanitized headers.

8.3.2. Contents of Linux

Installed files: config-2.6.27.4, lfskernel-2.6.27.4, and System.map-2.6.27.4

Short Descriptions

`config-2.6.27.4` Contains all the configuration selections for the kernel

`lfskernel-2.6.27.4` The engine of the Linux system. When turning on the computer, the kernel is the first part of the operating system that gets loaded. It detects and initializes all components of the computer's hardware, then makes these components available as a tree of files to the software and turns a single CPU into a multitasking machine capable of running scores of programs seemingly at the same time

`System.map-2.6.27.4` A list of addresses and symbols; it maps the entry points and addresses of all the functions and data structures in the kernel

8.4. Making the LFS System Bootable

Your shiny new LFS system is almost complete. One of the last things to do is to ensure that the system can be properly booted. The instructions below apply only to computers of IA-32 architecture, meaning mainstream PCs. Information on “boot loading” for other architectures should be available in the usual resource-specific locations for those architectures.

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

Earlier, we compiled and installed the GRUB boot loader software in preparation for this step. The procedure involves writing some special GRUB files to specific locations on the hard drive. We highly recommend creating a GRUB boot floppy diskette as a backup. Insert a blank floppy diskette and run the following commands:

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

Remove the diskette and store it somewhere safe. Now, run the **grub** shell:

```
grub
```

GRUB uses its own naming structure for drives and partitions in the form of *(hdn,m)*, where *n* is the hard drive number and *m* is the partition number, both starting from zero. For example, partition `hda1` is *(hd0,0)* to GRUB and `hdb3` is *(hd1,2)*. In contrast to Linux, GRUB does not consider CD-ROM drives to be hard drives. For example, if using a CD on `hdb` and a second hard drive on `hdc`, that second hard drive would still be *(hd1)*.

Using the above information, determine the appropriate designator for the root partition (or boot partition, if a separate one is used). For the following example, it is assumed that the root (or separate boot) partition is `hda4`.

Tell GRUB where to search for its `stage{1,2}` files. The Tab key can be used everywhere to make GRUB show the alternatives:

```
root (hd0,3)
```

Warning

The following command will overwrite the current boot loader. Do not run the command if this is not desired, for example, if using a third party boot manager to manage the Master Boot Record (MBR). In this scenario, it would make more sense to install GRUB into the “boot sector” of the LFS partition. In this case, this next command would become **`setup (hd0,3)`**.

Tell GRUB to install itself into the MBR of `hda`:

```
setup (hd0)
```

If all went well, GRUB will have reported finding its files in `/boot/grub`. That's all there is to it. Quit the **grub** shell:

```
quit
```

Create a “menu list” file defining GRUB's boot menu:

```
cat > /boot/grub/menu.lst << "EOF"
# Begin /boot/grub/menu.lst

# By default boot the first menu entry.
default 0

# Allow 30 seconds before booting the default.
timeout 30

# Use prettier colors.
color green/black light-green/black

# The first entry is for LFS.
title LFS 6.4
root (hd0,3)
kernel /boot/lfskernel-2.6.27.4 root=/dev/hda4
EOF
```

Add an entry for the host distribution if desired. It might look like this:

```
cat >> /boot/grub/menu.lst << "EOF"
title Red Hat
root (hd0,2)
kernel /boot/kernel-2.6.5 root=/dev/hda3
initrd /boot/initrd-2.6.5
EOF
```

If dual-booting Windows, the following entry will allow booting it:

```
cat >> /boot/grub/menu.lst << "EOF"
title Windows
rootnoverify (hd0,0)
chainloader +1
EOF
```

If **info grub** does not provide all necessary material, additional information regarding GRUB is located on its website at: *http://www.gnu.org/software/grub/*.

The FHS stipulates that GRUB's `menu.lst` file should be symlinked to `/etc/grub/menu.lst`. To satisfy this requirement, issue the following command:

```
mkdir -v /etc/grub
ln -sv /boot/grub/menu.lst /etc/grub
```

Chapter 9. The End

9.1. The End

Well done! The new LFS system is installed! We wish you much success with your shiny new custom-built Linux system.

It may be a good idea to create an `/etc/lfs-release` file. By having this file, it is very easy for you (and for us if you need to ask for help at some point) to find out which LFS version is installed on the system. Create this file by running:

```
echo 6.4 > /etc/lfs-release
```

9.2. Get Counted

Now that you have finished the book, do you want to be counted as an LFS user? Head over to *http://www.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.

9.3. Rebooting the System

Now that all of the software has been installed, it is time to reboot your computer. However, you should be aware of a few things. The system you have created in this book is quite minimal, and most likely will not have the functionality you would need to be able to continue forward. By installing a few extra packages from the BLFS book while still in our current chroot environment, you can leave yourself in a much better position to continue on once you reboot into your new LFS installation. Installing a text mode web browser, such as Lynx, you can easily view the BLFS book in one virtual terminal, while building packages in another. The GPM package will also allow you to perform copy/paste actions in your virtual terminals. Lastly, if you are in a situation where static IP configuration does not meet your networking requirements, installing packages such as Dhcpcd or PPP at this point might also be useful.

Now that we have said that, lets move on to booting our shiny new LFS installation for the first time! First exit from the chroot environment:

```
logout
```

Then unmount the virtual files systems:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Unmount the LFS file system itself:

```
umount -v $LFS
```

If multiple partitions were created, unmount the other partitions before unmounting the main one, like this:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Now, reboot the system with:

```
shutdown -r now
```

Assuming the GRUB boot loader was set up as outlined earlier, the menu is set to boot *LFS 6.4* automatically.

When the reboot is complete, the LFS system is ready for use and more software may be added to suit your needs.

9.4. What Now?

Thank you for reading this LFS book. We hope that you have found this book helpful and have learned more about the system creation process.

Now that the LFS system is installed, you may be wondering “What next?” To answer that question, we have compiled a list of resources for you.

- Maintenance

  Bugs and security notices are reported regularly for all software. Since an LFS system is compiled from source, it is up to you to keep abreast of such reports. There are several online resources that track such reports, some of which are shown below:

  - Freshmeat.net (*http://freshmeat.net/*)

    Freshmeat can notify you (via email) of new versions of packages installed on your system.

  - *CERT* (Computer Emergency Response Team)

    CERT has a mailing list that publishes security alerts concerning various operating systems and applications. Subscription information is available at *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq is a full-disclosure computer security mailing list. It publishes newly discovered security issues, and occasionally potential fixes for them. Subscription information is available at *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

  The Beyond Linux From Scratch book covers installation procedures for a wide range of software beyond the scope of the LFS Book. The BLFS project is located at *http://www.linuxfromscratch.org/blfs/*.

- LFS Hints

  The LFS Hints are a collection of educational documents submitted by volunteers in the LFS community. The hints are available at *http://www.linuxfromscratch.org/hints/list.html*.

- Mailing lists

  There are several LFS mailing lists you may subscribe to if you are in need of help, want to stay current with the latest developments, want to contribute to the project, and more. See Chapter 1 - Mailing Lists for more information.

- The Linux Documentation Project

  The goal of The Linux Documentation Project (TLDP) is to collaborate on all of the issues of Linux documentation. The TLDP features a large collection of HOWTOs, guides, and man pages. It is located at *http://www.tldp.org/*.

Part IV. Appendices

Appendix A. Acronyms and Terms

ABI	Application Binary Interface
ALFS	Automated Linux From Scratch
ALSA	Advanced Linux Sound Architecture
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input/Output System
BLFS	Beyond Linux From Scratch
BSD	Berkeley Software Distribution
chroot	change root
CMOS	Complementary Metal Oxide Semiconductor
COS	Class Of Service
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CVS	Concurrent Versions System
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
EGA	Enhanced Graphics Adapter
ELF	Executable and Linkable Format
EOF	End of File
EQN	equation
EVMS	Enterprise Volume Management System
ext2	second extended file system
ext3	third extended file system
FAQ	Frequently Asked Questions
FHS	Filesystem Hierarchy Standard
FIFO	First-In, First Out
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GB	Gibabytes
GCC	GNU Compiler Collection
GID	Group Identifier
GMT	Greenwich Mean Time
GPG	GNU Privacy Guard
HTML	Hypertext Markup Language

IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronic Engineers
IO	Input/Output
IP	Internet Protocol
IPC	Inter-Process Communication
IRC	Internet Relay Chat
ISO	International Organization for Standardization
ISP	Internet Service Provider
KB	Kilobytes
LED	Light Emitting Diode
LFS	Linux From Scratch
LSB	Linux Standard Base
MB	Megabytes
MBR	Master Boot Record
MD5	Message Digest 5
NIC	Network Interface Card
NLS	Native Language Support
NNTP	Network News Transport Protocol
NPTL	Native POSIX Threading Library
OSS	Open Sound System
PCH	Pre-Compiled Headers
PCRE	Perl Compatible Regular Expression
PID	Process Identifier
PLFS	Pure Linux From Scratch
PTY	pseudo terminal
QA	Quality Assurance
QOS	Quality Of Service
RAM	Random Access Memory
RPC	Remote Procedure Call
RTC	Real Time Clock
SBU	Standard Build Unit
SCO	The Santa Cruz Operation
SGR	Select Graphic Rendition
SHA1	Secure-Hash Algorithm 1
SMP	Symmetric Multi-Processor
TLDP	The Linux Documentation Project

TFTP	Trivial File Transfer Protocol
TLS	Thread-Local Storage
UID	User Identifier
umask	user file-creation mask
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
VC	Virtual Console
VGA	Video Graphics Array
VT	Virtual Terminal

Appendix B. Acknowledgments

We would like to thank the following people and organizations for their contributions to the Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – LFS Creator, LFS Project Leader
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – LFS Release Manager
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – LFS/BLFS/HLFS XML and XSL Maintainer
- *Jim Gifford* <jim@linuxfromscratch.org> – CLFS Project Co-Leader
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – LFS Technical Writer
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – LFS Technical Writer, LFS LiveCD Maintainer
- *Randy McMurchy* <randy@linuxfromscratch.org> – BLFS Project Leader, LFS Editor
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – LFS and BLFS Editor
- *DJ Lucas* <dj@linuxfromscratch.org> – LFS and BLFS Editor
- *Ken Moffat* <ken@linuxfromscratch.org> – LFS and CLFS Editor
- *Ryan Oliver* <ryan@linuxfromscratch.org> – CLFS Project Co-Leader
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – LFS Technical Writer, LFS Internationalization Editor, LFS Live CD Maintainer
- Countless other people on the various LFS and BLFS mailing lists who helped make this book possible by giving their suggestions, testing the book, and submitting bug reports, instructions, and their experiences with installing various packages.

Translators

- *Manuel Canales Esparcia* <macana@macana-es.com> – Spanish LFS translation project
- *Johan Lenglet* <johan@linuxfromscratch.org> – French LFS translation project
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Portuguese LFS translation project
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – German LFS translation project

Mirror Maintainers

North American Mirrors

- *Scott Kveton* <scott@osuosl.org> – lfs.oregonstate.edu mirror
- *William Astle* <lost@l-w.net> – ca.linuxfromscratch.org mirror
- *Eujon Sellers* <jpolen@rackspace.com> – lfs.introspeed.com mirror
- *Justin Knierim* <tim@idge.net> – lfs-matrix.net mirror

South American Mirrors

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info mirror

- *Luis Falcon* <Luis Falcon> – torredehanoi.org mirror

European Mirrors

- *Guido Passet* <guido@primerelay.net> – nl.linuxfromscratch.org mirror
- *Bastiaan Jacques* <baafie@planet.nl> – lfs.pagefault.net mirror
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – lfs.lineo.be mirror
- Scarlet Belgium – lfs.scarlet.be mirror
- *Sebastian Faulborn* <info@aliensoft.org> – lfs.aliensoft.org mirror
- *Stuart Fox* <stuart@dontuse.ms> – lfs.dontuse.ms mirror
- *Ralf Uhlemann* <admin@realhost.de> – lfs.oss-mirror.org mirror
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – at.linuxfromscratch.org mirror
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – se.linuxfromscratch.org mirror
- *Franck* <franck@linuxpourtous.com> – lfs.linuxpourtous.com mirror
- *Philippe Baqué* <baque@cict.fr> – lfs.cict.fr mirror
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – lfs.pilgrims.ru mirror
- *Benjamin Heil* <kontakt@wankoo.org> – lfs.wankoo.org mirror

Asian Mirrors

- *Satit Phermsawang* <satit@wbac.ac.th> – lfs.phayoune.org mirror
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – lfs.mirror.shizu-net.jp mirror
- *Init World* <http://www.initworld.com/> – lfs.initworld.com mirror

Australian Mirrors

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org mirror

Former Project Team Members

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – LFS Book Editor
- Archaic <archaic@linuxfromscratch.org> – LFS Technical Writer/Editor, HLFS Project Leader, BLFS Editor, Hints and Patches Project Maintainer
- *Nathan Coulson* <nathan@linuxfromscratch.org> – LFS-Bootscripts Maintainer
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Website Developer, FAQ Maintainer
- Alex Groenewoud – LFS Technical Writer
- Marc Heerdink
- Mark Hymers
- Seth W. Klein – FAQ maintainer

- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Wiki Maintainer
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Website Backend-Scripts Maintainer
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – LFS NNTP Gateway Maintainer
- *Greg Schafer* <gschafer@zip.com.au> – LFS Technical Writer
- Jesse Tie-Ten-Quee – LFS Technical Writer
- *James Robertson* <jwrober@linuxfromscratch.org> – Bugzilla Maintainer
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – BLFS Book Editor, Hints and Patches Project Leader
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer
- *Zack Winkles* <zwinkles@gmail.com> – LFS Technical Writer

A very special thank you to our donators

- *Dean Benson* <dean@vipersoft.co.uk> for several monetary contributions
- *Hagen Herrschaft* <hrx@hrxnet.de> for donating a 2.2 GHz P4 system, now running under the name of Lorien
- *SEO Company Canada* supports Open Source projects and different Linux distributions
- *VA Software* who, on behalf of *Linux.com*, donated a VA Linux 420 (former StartX SP2) workstation
- Mark Stone for donating Belgarath, the first linuxfromscratch.org server

Appendix C. Dependencies

Every package built in LFS relies on one or more other packages in order to build and install properly. Some packages even participate in circular dependencies, that is, the first package depends on the second which in turn depends on the first. Because of these dependencies, the order in which packages are built in LFS is very important. The purpose of this page is to document the dependencies of each package built in LFS.

For each package we build, we have listed three types of dependencies. The first lists what other packages need to be available in order to compile and install the package in question. The second lists what packages, in addition to those on the first list, need to be available in order to run the testsuites. The last list of dependencies are packages that require this package to be built and installed in its final location before they are built and installed. In most cases, this is because these packages will hardcode paths to binaries within their scripts. If not built in a certain order, this could result in paths of /tools/bin/[binary] being placed inside scripts installed to the final system. This is obviously not desirable.

Autoconf

Installation depends on: Bash, Coreutils, Grep, M4, Make, Perl, Sed, and Texinfo

Test suite depends on: Automake, Diffutils, Findutils, GCC, and Libtool

Must be installed before: Automake

Automake

Installation depends on: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo

Test suite depends on: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, and Tar. Can also use several other packages that are not installed in LFS.

Must be installed before: None

Bash

Installation depends on: Bash, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, and Texinfo

Test suite depends on: None

Must be installed before: None

Berkeley DB

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, and Sed

Test suite depends on: Not run. Requires TCL installed on the final system

Must be installed before: None

Binutils

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl, Sed, and Texinfo

Test suite depends on: DejaGNU and Expect

Must be installed before: None

Bison

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, and Sed
Test suite depends on:	Diffutils and Findutils
Must be installed before:	Flex, Kbd, and Tar

Bzip2

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, and Patch
Test suite depends on:	None
Must be installed before:	None

Coreutils

Installation depends on:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, and Texinfo
Test suite depends on:	Diffutils, E2fsprogs
Must be installed before:	Bash, Diffutils, Findutils, Man-DB, and Udev

DejaGNU

Installation depends on:	Bash, Coreutils, Diffutils, GCC, Grep, Make, and Sed
Test suite depends on:	No testsuite available
Must be installed before:	None

Diffutils

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, and Texinfo
Test suite depends on:	No testsuite available
Must be installed before:	None

Expect

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, and Tcl
Test suite depends on:	None
Must be installed before:	None

E2fsprogs

Installation depends on:	Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, Make, Sed, and Texinfo
Test suite depends on:	Diffutils
Must be installed before:	Util-Linux

File

Installation depends on:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Zlib
Test suite depends on:	No testsuite available
Must be installed before:	None

Findutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: DejaGNU, Diffutils, and Expect
Must be installed before: None

Flex

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, and Texinfo
Test suite depends on: Bison and Gawk
Must be installed before: IPRoute2, Kbd, and Man-DB

Gawk

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed and, Texinfo
Test suite depends on: Diffutils
Must be installed before: None

Gcc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP (Chapter 6), Grep, M4 (Chapter 5), Make, MPFR (Chapter 6), Patch, Perl, Sed, Tar, and Texinfo
Test suite depends on: DejaGNU and Expect
Must be installed before: None

Gettext

Installation depends on: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Diffutils, Perl, and Tcl
Must be installed before: Automake

Glibc

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed, and Texinfo
Test suite depends on: None
Must be installed before: None

GMP

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed and Texinfo
Test suite depends on: None
Must be installed before: MPFR, GCC

Grep

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, and Texinfo
Test suite depends on: Gawk
Must be installed before: Man-DB

Groff

Installation depends on: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, and Texinfo

Test suite depends on: No testsuite available

Must be installed before: Man-DB and Perl

GRUB

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Sed, and Texinfo

Test suite depends on: None

Must be installed before: None

Gzip

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo

Test suite depends on: Diffutils

Must be installed before: Man-DB

Iana-Etc

Installation depends on: Coreutils, Gawk, and Make

Test suite depends on: No testsuite available

Must be installed before: Perl

Inetutils

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, and Texinfo

Test suite depends on: No testsuite available

Must be installed before: Tar

IProute2

Installation depends on: Bash, Berkeley DB, Bison, Coreutils, Flex, GCC, Glibc, Make, and Linux API Headers

Test suite depends on: No testsuite available

Must be installed before: None

Kbd

Installation depends on: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, and Sed

Test suite depends on: No testsuite available

Must be installed before: None

Less

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed

Test suite depends on: No testsuite available

Must be installed before: None

Libtool

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Findutils
Must be installed before: None

Linux Kernel

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Module-Init-Tools, Ncurses, and Sed
Test suite depends on: No testsuite available
Must be installed before: None

M4

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Diffutils
Must be installed before: Autoconf and Bison

Make

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
Test suite depends on: Perl and Procps
Must be installed before: None

Man-DB

Installation depends on: Bash, Berkeley DB, Binutils, Bzip2, Coreutils, Flex, GCC, Gettext, Glibc, Grep, Groff, Gzip, Less, Make, and Sed
Test suite depends on: Not run. Requires Man-DB testsuite package
Must be installed before: None

Module-Init-Tools

Installation depends on: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Make, Patch, Sed, and Zlib
Test suite depends on: Diffutils, File, Gawk, Gzip, and Mktemp
Must be installed before: None

MPFR

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed and Texinfo
Test suite depends on: None
Must be installed before: GCC

Ncurses

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, and Sed
Test suite depends on: No testsuite available
Must be installed before: Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-Linux, and Vim

Patch

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, and Sed

Test suite depends on: No testsuite available

Must be installed before: None

Perl

Installation depends on: Bash, Berkeley DB, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Groff, Make, Sed, and Zlib

Test suite depends on: Iana-Etc and Procps

Must be installed before: Autoconf

Procps

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Make, and Ncurses

Test suite depends on: No testsuite available

Must be installed before: None

Psmisc

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, and Sed

Test suite depends on: No testsuite available

Must be installed before: None

Readline

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, and Texinfo

Test suite depends on: No testsuite available

Must be installed before: Bash

Sed

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo

Test suite depends on: Diffutils and Gawk

Must be installed before: E2fsprogs, File, Libtool, and Shadow

Shadow

Installation depends on: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, and Sed

Test suite depends on: No testsuite available

Must be installed before: None

Sysklogd

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make, and Patch

Test suite depends on: No testsuite available

Must be installed before: None

Sysvinit

Installation depends on: Binutils, Coreutils, GCC, Glibc, Make, and Sed
Test suite depends on: No testsuite available
Must be installed before: None

Tar

Installation depends on: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, and Texinfo
Test suite depends on: Diffutils, Findutils, Gawk, and Gzip
Must be installed before: None

Tcl

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, and Sed
Test suite depends on: None
Must be installed before: None

Texinfo

Installation depends on: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, and Sed
Test suite depends on: None
Must be installed before: None

Udev

Installation depends on: Binutils, Coreutils, GCC, Glibc, and Make
Test suite depends on: Findutils, Perl, and Sed
Must be installed before: None

Util-Linux

Installation depends on: Bash, Binutils, Coreutils, E2fprogs, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, Sed, and Zlib
Test suite depends on: No testsuite available
Must be installed before: None

Vim

Installation depends on: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
Test suite depends on: None
Must be installed before: None

Zlib

Installation depends on: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, and Sed
Test suite depends on: None
Must be installed before: File, Module-Init-Tools, Perl, and Util-Linux

Appendix D. Boot and sysconfig scripts version-20081031

The scripts in this appendix are listed by the directory where they normally reside. The order is `/etc/rc.d/init.d`, `/etc/sysconfig`, `/etc/sysconfig/network-devices`, and `/etc/sysconfig/network-devices/services`. Within each section, the files are listed in the order they are normally called.

D.1. /etc/rc.d/init.d/rc

The `rc` script is the first script called by init and initiates the boot process.

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/rc
#
# Description : Main Run Level Control Script
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# This sets a few default terminal options.
stty sane

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" = "" ]; then
    echo "Usage: ${0} <runlevel>" >&2
    exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" = "" ] && previous=N

if [ ! -d ${rc_base}/rc${runlevel}.d ]; then
    boot_mesg "${rc_base}/rc${runlevel}.d does not exist." ${WARNING}
    boot_mesg_flush
    exit 1
fi

# Attempt to stop all service started by previous runlevel,
# and killed in this runlevel
```

```
if [ "${previous}" != "N" ]; then
    for i in $(ls -v ${rc_base}/rc${runlevel}.d/K* 2> /dev/null)
    do
        check_script_status

        suffix=${i#$rc_base/rc$runlevel.d/K[0-9][0-9]}
        prev_start=$rc_base/rc$previous.d/S[0-9][0-9]$suffix
        sysinit_start=$rc_base/rcsysinit.d/S[0-9][0-9]$suffix

        if [ "${runlevel}" != "0" ] && [ "${runlevel}" != "6" ]; then
            if [ ! -f ${prev_start} ] && [ ! -f ${sysinit_start} ]; then
                boot_mesg -n "WARNING:\n\n${i} can't be" ${WARNING}
                boot_mesg -n " executed because it was not"
                boot_mesg -n " not started in the previous"
                boot_mesg -n " runlevel (${previous})."
                boot_mesg "" ${NORMAL}
                boot_mesg_flush
                continue
            fi
        fi
        ${i} stop
        error_value=${?}

        if [ "${error_value}" != "0" ]; then
            print_error_msg
        fi
    done
fi

#Start all functions in this runlevel
for i in $( ls -v ${rc_base}/rc${runlevel}.d/S* 2> /dev/null)
do
    if [ "${previous}" != "N" ]; then
        suffix=${i#$rc_base/rc$runlevel.d/S[0-9][0-9]}
        stop=$rc_base/rc$runlevel.d/K[0-9][0-9]$suffix
        prev_start=$rc_base/rc$previous.d/S[0-9][0-9]$suffix

        [ -f ${prev_start} ] && [ ! -f ${stop} ] && continue
    fi

    check_script_status

    case ${runlevel} in
        0|6)
            ${i} stop
            ;;
        *)
            ${i} start
            ;;
    esac
    error_value=${?}

    if [ "${error_value}" != "0" ]; then
        print_error_msg
    fi
done
```

```
# End $rc_base/init.d/rc
```

D.2. /etc/rc.d/init.d/functions

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/functions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#        http://winterdrache.de/linux/newboot/index.html
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

# Signal sent to running processes to refresh their configuration
RELOADSIG="HUP"

# Number of seconds between STOPSIG and FALLBACK when stopping processes
KILLDELAY="3"

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
    COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## Provide an echo that supports -e and -n
# If formatting is needed, $ECHO should be used
case "`echo -e -n test`" in
    -[en]*)
        ECHO=/bin/echo
        ;;
    *)
        ECHO=echo
        ;;
esac
```

```
## Set Cursor Position Commands, used via $ECHO
SET_COL="\\033[${COL}G"      # at the $COL char
SET_WCOL="\\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\\033[1A\\033[0G"   # Up one line, at the 0'th char

## Set color commands, used via $ECHO
# Please consult `man console_codes for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles
NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

STRING_LENGTH="0"   # the length of the current message

#*******************************************************************************
# Function - boot_mesg()
#
# Purpose:      Sending information from bootup scripts to the console
#
# Inputs:       $1 is the message
#               $2 is the colorcode for the console
#
# Outputs:      Standard Output
#
# Dependencies: - sed for parsing strings.
#            - grep for counting string length.
#
# Todo:
#*******************************************************************************
boot_mesg()
{
    local ECHOPARM=""

    while true
    do
        case "${1}" in
            -n)
                ECHOPARM=" -n "
                shift 1
                ;;
            -*)
                echo "Unknown Option: ${1}"
                return 1
                ;;
            *)
                break
                ;;
        esac
```

```
    done

    ## Figure out the length of what is to be printed to be used
    ## for warning messages.
    STRING_LENGTH=$((${#1} + 1))

    # Print the message to the screen
    ${ECHO} ${ECHOPARM} -e "${2}${1}"

}

boot_mesg_flush()
{
    # Reset STRING_LENGTH for next message
    STRING_LENGTH="0"
}

boot_log()
{
    # Left in for backwards compatibility
    :
}

echo_ok()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${SUCCESS}  OK  ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
        boot_mesg_flush
}

echo_failure()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${FAILURE} FAIL ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
        boot_mesg_flush
}

echo_warning()
{
    ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} WARN ${BRACKET}]"
    ${ECHO} -e "${NORMAL}"
        boot_mesg_flush
}

print_error_msg()
{
    echo_failure
    # $i is inherited by the rc script
    boot_mesg -n "FAILURE:\n\nYou should not be reading this error message.\n\n" ${FAILURE}
    boot_mesg -n " It means that an unforeseen error took"
    boot_mesg -n " place in ${i}, which exited with a return value of"
    boot_mesg " ${error_value}.\n"
    boot_mesg_flush
    boot_mesg -n "If you're able to track this"
    boot_mesg -n " error down to a bug in one of the files provided by"
    boot_mesg -n " the LFS book, please be so kind to inform us at"
    boot_mesg " lfs-dev@linuxfromscratch.org.\n"
```

```
    boot_mesg_flush
    boot_mesg -n "Press Enter to continue..." ${INFO}
    boot_mesg "" ${NORMAL}
    read ENTER
}

check_script_status()
{
    # $i is inherited by the rc script
    if [ ! -f ${i} ]; then
        boot_mesg "${i} is not a valid symlink." ${WARNING}
        echo_warning
        continue
    fi

    if [ ! -x ${i} ]; then
        boot_mesg "${i} is not executable, skipping." ${WARNING}
        echo_warning
        continue
    fi
}

evaluate_retval()
{
    error_value="${?}"

    if [ ${error_value} = 0 ]; then
        echo_ok
    else
        echo_failure
    fi

    # This prevents the 'An Unexpected Error Has Occurred' from trivial
    # errors.
    return 0
}

print_status()
{
    if [ "${#}" = "0" ]; then
        echo "Usage: ${0} {success|warning|failure}"
        return 1
    fi

    case "${1}" in

        success)
            echo_ok
            ;;

        warning)
            # Leave this extra case in because old scripts
            # may call it this way.
            case "${2}" in
                running)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
```

```
                    boot_mesg "Already running." ${WARNING}
                    echo_warning
                    ;;
                not_running)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Not running." ${WARNING}
                    echo_warning
                    ;;
                not_available)
                    ${ECHO} -e -n "${CURS_UP}"
                    ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                    boot_mesg "Not available." ${WARNING}
                    echo_warning
                    ;;
                *)
                    # This is how it is supposed to
                    # be called
                    echo_warning
                    ;;
            esac
        ;;

        failure)
            echo_failure
        ;;

    esac

}

reloadproc()
{
    local pidfile=""
    local failure=0

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" -lt "1" ]; then
        log_failure_msg "Usage: reloadproc [-p pidfile] pathname"
        return 2
    fi
```

```
    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi

    # Is the process running?
    if [ -z "${pidfile}" ];    then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Warn about stale pid file
    if [ "$?" = 1 ]; then
        boot_mesg -n "Removing stale pid file: ${pidfile}. " ${WARNING}
        rm -f "${pidfile}"
    fi

    if [ -n "${pidlist}" ];    then
        for pid in ${pidlist}
        do
            kill -"${RELOADSIG}" "${pid}" || failure="1"
        done

        (exit ${failure})
        evaluate_retval

    else
        boot_mesg "Process ${1} not running." ${WARNING}
        echo_warning
    fi
}

statusproc()
{
    local pidfile=""
    local base=""
    local ret=""

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done
```

```
    if [ "${#}" != "1" ]; then
        shift 1
        log_failure_msg "Usage: statusproc [-p pidfile] pathname"
        return 2
    fi

    # Get the process basename
    base="${1##*/}"

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi

    # Is the process running?
    if [ -z "${pidfile}" ];    then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Store the return status
    ret=$?

    if [ -n "${pidlist}" ];    then
        ${ECHO} -e "${INFO}${base} is running with Process"\
            "ID(s) ${pidlist}.${NORMAL}"
    else
        if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
            ${ECHO} -e "${WARNING}${1} is not running but"\
                "/var/run/${base}.pid exists.${NORMAL}"
        else
            if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
                ${ECHO} -e "${WARNING}${1} is not running"\
                    "but ${pidfile} exists.${NORMAL}"
            else
                ${ECHO} -e "${INFO}${1} is not running.${NORMAL}"
            fi
        fi
    fi

    # Return the status from pidofproc
    return $ret
}

# The below functions are documented in the LSB-generic 2.1.0

#*******************************************************************************
# Function - pidofproc [-s] [-p pidfile] pathname
#
# Purpose: This function returns one or more pid(s) for a particular daemon
#
# Inputs: -p pidfile, use the specified pidfile instead of pidof
#         pathname, path to the specified program
#
# Outputs: return 0 - Success, pid's in stdout
#          return 1 - Program is dead, pidfile exists
```

```
#           return 2 - Invalid or excessive number of arguments,
#                      warning in stdout
#           return 3 - Program is not running
#
# Dependencies: pidof, echo, head
#
# Todo: Remove dependency on head
#       This depreciates getpids
#       Test changes to pidof
#
#*******************************************************************************
pidofproc()
{
    local pidfile=""
    local lpids=""
    local silent=""
    pidlist=""
    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;

            -s)
                # Added for legacy opperation of getpids
                # eliminates several '> /dev/null'
                silent="1"
                shift 1
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" != "1" ]; then
        shift 1
        log_failure_msg "Usage: pidofproc [-s] [-p pidfile] pathname"
        return 2
    fi

    if [ -n "${pidfile}" ]; then
        if [ ! -r "${pidfile}" ]; then
            return 3 # Program is not running
        fi

        lpids=`head -n 1 ${pidfile}`
        for pid in ${lpids}
        do
            if [ "${pid}" -ne "$$" -a "${pid}" -ne "${PPID}" ]; then
                kill -0 "${pid}" 2>/dev/null &&
```

```
                pidlist="${pidlist} ${pid}"
            fi

            if [ "${silent}" != "1" ]; then
                echo "${pidlist}"
            fi

            test -z "${pidlist}" &&
            # Program is dead, pidfile exists
            return 1
            # else
            return 0
        done

    else
        pidlist=`pidof -o $$ -o $PPID -x "$1"`
        if [ "${silent}" != "1" ]; then
            echo "${pidlist}"
        fi

        # Get provide correct running status
        if [ -n "${pidlist}" ]; then
            return 0
        else
            return 3
        fi

    fi

    if [ "$?" != "0" ]; then
        return 3 # Program is not running
    fi
}

# This will ensure compatibility with previous LFS Bootscripts
getpids()
{
    if [ -z "${PIDFILE}" ]; then
        pidofproc -s -p "${PIDFILE}" $@
    else
        pidofproc -s $@
    fi
    base="${1##*/}"
}

#*******************************************************************************
# Function - loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]
#
# Purpose: This runs the specified program as a daemon
#
# Inputs: -f, run the program even if it is already running
#         -n nicelevel, specifies a nice level. See nice(1).
#         -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         args, arguments to pass to specified program
#
# Outputs: return 0 - Success
```

```
#           return 2 - Invalid of excessive number of arguments,
#                      warning in stdout
#           return 4 - Program or service status is unknown
#
# Dependencies: nice, rm
#
# Todo: LSB says this should be called start_daemon
#       LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#       loadproc returns 0 if program is already running, not LSB compliant
#
#********************************************************************************
loadproc()
{
    local pidfile=""
    local forcestart=""
    local nicelevel="10"

# This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
        pidfile="${PIDFILE}"
    fi

  while true
    do
        case "${1}" in
            -f)
                forcestart="1"
                shift 1
                ;;
            -n)
                nicelevel="${2}"
                shift 2
                ;;
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2 #invalid or excess argument(s)
                ;;
            *)
                break
                ;;
        esac
    done

    if [ "${#}" = "0" ]; then
        log_failure_msg "Usage: loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]"
        return 2 #invalid or excess argument(s)
    fi

    if [ -z "${forcestart}" ]; then
        if [ -z "${pidfile}" ];    then
            pidofproc -s "${1}"
```

```
        else
            pidofproc -s -p "${pidfile}" "${1}"
        fi

        case "${?}" in
            0)
                log_warning_msg "Unable to continue: ${1} is running"
                return 0 # 4
                ;;
            1)
                boot_mesg "Removing stale pid file: ${pidfile}" ${WARNING}
                rm -f "${pidfile}"
                ;;
            3)
                ;;
            *)
                log_failure_msg "Unknown error code from pidofproc: ${?}"
                return 4
                ;;
        esac
    fi

    nice -n "${nicelevel}" "${@}"
    evaluate_retval # This is "Probably" not LSB compliant,
#                         but required to be compatible with older bootscripts
    return 0
}

#*******************************************************************************
# Function - killproc  [-p pidfile] pathname [signal]
#
# Purpose:
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Unknown Status
#
# Dependencies: kill, rm
#
# Todo: LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#
#*******************************************************************************
killproc()
{
    local pidfile=""
    local killsig=TERM # default signal is SIGTERM
    pidlist=""

    # This will ensure compatibility with previous LFS Bootscripts
    if [ -n "${PIDFILE}" ];    then
```

```
        pidfile="${PIDFILE}"
    fi

    while true
    do
        case "${1}" in
            -p)
                pidfile="${2}"
                shift 2
                ;;
            -*)
                log_failure_msg "Unknown Option: ${1}"
                return 2
                ;;
            *)
                 break
                ;;
        esac
    done

    if [ "${#}" = "2" ]; then
        killsig="${2}"
    elif [ "${#}" != "1" ];    then
        shift 2
        log_failure_msg "Usage: killproc  [-p pidfile] pathname [signal]"
        return 2
    fi

    # Is the process running?
    if [ -z "${pidfile}" ];    then
        pidofproc -s "${1}"
    else
        pidofproc -s -p "${pidfile}" "${1}"
    fi

    # Remove stale pidfile
    if [ "$?" = 1 ]; then
        boot_mesg "Removing stale pid file: ${pidfile}." ${WARNING}
        rm -f "${pidfile}"
    fi

    # If running, send the signal
    if [ -n "${pidlist}" ]; then
    for pid in ${pidlist}
    do
        kill -${killsig} ${pid} 2>/dev/null

        # Wait up to 3 seconds, for ${pid} to terminate
        case "${killsig}" in
        TERM|SIGTERM|KILL|SIGKILL)
            # sleep in 1/10ths of seconds and
            # multiply KILLDELAY by 10
            local dtime="${KILLDELAY}0"
            while [ "${dtime}" != "0" ]
            do
                kill -0 ${pid} 2>/dev/null || break
                sleep 0.1
```

```
                dtime=$(( ${dtime} - 1))
            done
            # If ${pid} is still running, kill it
            kill -0 ${pid} 2>/dev/null && kill -KILL ${pid} 2>/dev/null
            ;;
        esac
    done

    # Check if the process is still running if we tried to stop it
    case "${killsig}" in
    TERM|SIGTERM|KILL|SIGKILL)
        if [ -z "${pidfile}" ];    then
            pidofproc -s "${1}"
        else
            pidofproc -s -p "${pidfile}" "${1}"
        fi

        # Program was terminated
        if [ "$?" != "0" ]; then
            # Remove the pidfile if necessary
            if [ -f "${pidfile}" ];    then
                rm -f "${pidfile}"
            fi
            echo_ok
            return 0
        else # Program is still running
            echo_failure
            return 4 # Unknown Status
        fi
        ;;
    *)
        # Just see if the kill returned successfully
        evaluate_retval
        ;;
    esac
    else # process not running
    print_status warning not_running
    fi
}


#*******************************************************************************
# Function - log_success_msg "message"
#
# Purpose: Print a success message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*******************************************************************************
log_success_msg()
{
```

```
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""[""${SUCCESS}""  OK  ""${BRACKET}""]""${NORMAL}"
    return 0
}

#*******************************************************************************
# Function - log_failure_msg "message"
#
# Purpose: Print a failure message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*******************************************************************************
log_failure_msg() {
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""[""${FAILURE}"" FAIL ""${BRACKET}""]""${NORMAL}"
    return 0
}

#*******************************************************************************
# Function - log_warning_msg "message"
#
# Purpose: print a warning message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*******************************************************************************
log_warning_msg() {
    ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
    ${ECHO} -e "${SET_COL}""${BRACKET}""[""${WARNING}"" WARN ""${BRACKET}""]""${NORMAL}"
    return 0
}

# End $rc_base/init.d/functions
```

D.3. /etc/rc.d/init.d/mountkernfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/mountkernfs
#
# Description : Mount proc and sysfs
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
```

```
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg -n "Mounting kernel-based file systems:" ${INFO}

        if ! mountpoint /proc >/dev/null; then
            boot_mesg -n " /proc" ${NORMAL}
            mount -n /proc || failed=1
        fi

        if ! mountpoint /sys >/dev/null; then
            boot_mesg -n " /sys" ${NORMAL}
            mount -n /sys || failed=1
        fi

        boot_mesg "" ${NORMAL}

        (exit ${failed})
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/mountkernfs
```

D.4. /etc/rc.d/init.d/consolelog

```
#!/bin/sh
# Begin $rc_base/init.d/consolelog

########################################################################
#
# Description : Set the kernel log level for the console
#
# Authors     : Dan Nicholson - dnicholson@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       : /proc must be mounted before this can run
#
########################################################################

. /etc/sysconfig/rc
```

```
. ${rc_functions}

# set the default loglevel
LOGLEVEL=7
if [ -r /etc/sysconfig/console ]; then
    . /etc/sysconfig/console
fi

case "${1}" in
    start)
        case "$LOGLEVEL" in
        [1-8])
            boot_mesg "Setting the console log level to ${LOGLEVEL}..."
            dmesg -n $LOGLEVEL
            evaluate_retval
            ;;
        *)
            boot_mesg "Console log level '${LOGLEVEL}' is invalid" ${FAILURE}
            echo_failure
            ;;
        esac
        ;;
    status)
        # Read the current value if possible
        if [ -r /proc/sys/kernel/printk ]; then
            read level line < /proc/sys/kernel/printk
        else
            boot_mesg "Can't read the current console log level" ${FAILURE}
            echo_failure
        fi

        # Print the value
        if [ -n "$level" ]; then
            ${ECHO} -e "${INFO}The current console log level" \
                "is ${level}${NORMAL}"
        fi
        ;;

    *)
        echo "Usage: ${0} {start|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/consolelog
```

D.5. /etc/rc.d/init.d/modules

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/modules
#
# Description : Module auto-loading script
#
# Authors     : Zack Winkles
#
```

```
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Assure that the kernel has module support.
[ -e /proc/ksyms -o -e /proc/modules ] || exit 0

case "${1}" in
    start)

        # Exit if there's no modules file or there are no
        # valid entries
        [ -r /etc/sysconfig/modules ] &&
            egrep -qv '^($|#)' /etc/sysconfig/modules ||
            exit 0

        boot_mesg -n "Loading modules:" ${INFO}

        # Only try to load modules if the user has actually given us
        # some modules to load.
        while read module args; do

            # Ignore comments and blank lines.
            case "$module" in
                ""|"#"*) continue ;;
            esac

            # Attempt to load the module, making
            # sure to pass any arguments provided.
            modprobe ${module} ${args} >/dev/null

            # Print the module name if successful,
            # otherwise take note.
            if [ $? -eq 0 ]; then
                boot_mesg -n " ${module}" ${NORMAL}
            else
                failedmod="${failedmod} ${module}"
            fi
        done < /etc/sysconfig/modules

        boot_mesg "" ${NORMAL}
        # Print a message about successfully loaded
        # modules on the correct line.
        echo_ok

        # Print a failure message with a list of any
        # modules that may have failed to load.
        if [ -n "${failedmod}" ]; then
            boot_mesg "Failed to load modules:${failedmod}" ${FAILURE}
            echo_failure
        fi
        ;;
```

```
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/modules
```

D.6. /etc/rc.d/init.d/udev

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/udev
#
# Description : Udev cold-plugging script
#
# Authors     : Zack Winkles, Alexander E. Patrakov
#
# Version     : 00.02
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Populating /dev with device nodes..."
        if ! grep -q '[[:space:]]sysfs' /proc/mounts; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nUnable to create" ${FAILURE}
            boot_mesg -n " devices without a SysFS filesystem"
            boot_mesg -n "\n\nAfter you press Enter, this system"
            boot_mesg -n " will be halted and powered off."
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            /etc/rc.d/init.d/halt stop
        fi

        # Mount a temporary file system over /dev, so that any devices
        # made or removed during this boot don't affect the next one.
        # The reason we don't write to mtab is because we don't ever
        # want /dev to be unavailable (such as by `umount -a').
        mount -n -t tmpfs tmpfs /dev -o mode=755
        if [ ${?} != 0 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nCannot mount a tmpfs" ${FAILURE}
            boot_mesg -n " onto /dev, this system will be halted."
            boot_mesg -n "\n\nAfter you press Enter, this system"
            boot_mesg -n " will be halted and powered off."
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
```

```
            /etc/rc.d/init.d/halt stop
        fi

        # Udev handles uevents itself, so we don't need to have
        # the kernel call out to any binary in response to them
        echo > /proc/sys/kernel/hotplug

        # Copy static device nodes to /dev
        cp -a /lib/udev/devices/* /dev

        # Start the udev daemon to continually watch for, and act on,
        # uevents
        /sbin/udevd --daemon

        # Now traverse /sys in order to "coldplug" devices that have
        # already been discovered
        /sbin/udevadm trigger

        # Now wait for udevd to process the uevents we triggered
        /sbin/udevadm settle
        evaluate_retval

        ;;

    *)
        echo "Usage ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/udev
```

D.7. /etc/rc.d/init.d/swap

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/swap
#
# Description : Swap Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Activating all swap files/partitions..."
        swapon -a
        evaluate_retval
```

```
        ;;

    stop)
        boot_mesg "Deactivating all swap files/partitions..."
        swapoff -a
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        boot_mesg "Retrieving swap status." ${INFO}
        echo_ok
        echo
        swapon -s
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/swap
```

D.8. /etc/rc.d/init.d/setclock

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/setclock
#
# Description : Setting Linux Clock
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. /etc/sysconfig/clock

CLOCKPARAMS=

case "${UTC}" in
    yes|true|1)
        CLOCKPARAMS="${CLOCKPARAMS} --utc"
        ;;
```

```
    no|false|0)
        CLOCKPARAMS="${CLOCKPARAMS} --localtime"
        ;;

esac

case ${1} in
    start)
        boot_mesg "Setting system clock..."
        hwclock --hctosys ${CLOCKPARAMS} >/dev/null
        evaluate_retval
        ;;

    stop)
        boot_mesg "Setting hardware clock..."
        hwclock --systohc ${CLOCKPARAMS} >/dev/null
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        ;;

esac
```

D.9. /etc/rc.d/init.d/checkfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/checkfs
#
# Description : File System Check
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#        A. Luebke - luebke@users.sourceforge.net
#
# Version     : 00.00
#
# Notes       :
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0    - No errors
# 1    - File system errors corrected
# 2    - System should be rebooted
# 4    - File system errors left uncorrected
# 8    - Operational error
# 16   - Usage or syntax error
# 32   - Fsck canceled by user request
# 128  - Shared library error
#
#########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
```

```
case "${1}" in
    start)
        if [ -f /fastboot ]; then
            boot_mesg -n "/fastboot found, will not perform" ${INFO}
            boot_mesg " file system checks as requested."
            echo_ok
            exit 0
        fi

        boot_mesg "Mounting root file system in read-only mode..."
        mount -n -o remount,ro / >/dev/null
        evaluate_retval

        if [ ${?} != 0 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nCannot check root" ${FAILURE}
            boot_mesg -n " filesystem because it could not be mounted"
            boot_mesg -n " in read-only mode.\n\nAfter you"
            boot_mesg -n " press Enter, this system will be"
            boot_mesg -n " halted and powered off."
            boot_mesg -n "\n\nPress enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            ${rc_base}/init.d/halt stop
        fi

        if [ -f /forcefsck ]; then
            boot_mesg -n "/forcefsck found, forcing file" ${INFO}
            boot_mesg " system checks as requested."
            echo_ok
            options="-f"
        else
            options=""
        fi

        boot_mesg "Checking file systems..."
        # Note: -a option used to be -p; but this fails e.g.
        # on fsck.minix
        fsck ${options} -a -A -C -T
        error_value=${?}

        if [ "${error_value}" = 0 ]; then
            echo_ok
        fi

        if [ "${error_value}" = 1 ]; then
            echo_warning
            boot_mesg -n "WARNING:\n\nFile system errors" ${WARNING}
            boot_mesg -n " were found and have been corrected."
            boot_mesg -n "  You may want to double-check that"
            boot_mesg -n " everything was fixed properly."
            boot_mesg "" ${NORMAL}
        fi

        if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
            echo_warning
```

```
            boot_mesg -n "WARNING:\n\nFile system errors" ${WARNING}
            boot_mesg -n " were found and have been been"
             boot_mesg -n " corrected, but the nature of the"
            boot_mesg -n " errors require this system to be"
            boot_mesg -n " rebooted.\n\nAfter you press enter,"
            boot_mesg -n " this system will be rebooted"
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
            reboot -f
        fi

        if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nFile system errors" ${FAILURE}
            boot_mesg -n " were encountered that could not be"
            boot_mesg -n " fixed automatically.  This system"
            boot_mesg -n " cannot continue to boot and will"
            boot_mesg -n " therefore be halted until those"
            boot_mesg -n " errors are fixed manually by a"
            boot_mesg -n " System Administrator.\n\nAfter you"
            boot_mesg -n " press Enter, this system will be"
            boot_mesg -n " halted and powered off."
            boot_mesg -n "\n\nPress Enter to continue..." ${INFO}
            boot_mesg "" ${NORMAL}
            read ENTER
          ${rc_base}/init.d/halt stop
        fi

        if [ "${error_value}" -ge 16 ]; then
            echo_failure
            boot_mesg -n "FAILURE:\n\nUnexpected Failure" ${FAILURE}
            boot_mesg -n " running fsck.  Exited with error"
            boot_mesg -n " code: ${error_value}."
            boot_mesg "" ${NORMAL}
            exit ${error_value}
        fi
        ;;
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/checkfs
```

D.10. /etc/rc.d/init.d/mountfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/mountfs
#
# Description : File System Mount Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
```

```
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Remounting root file system in read-write mode..."
        mount -n -o remount,rw / >/dev/null
        evaluate_retval

        # Remove fsck-related file system watermarks.
        rm -f /fastboot /forcefsck

        boot_mesg "Recording existing mounts in /etc/mtab..."
        > /etc/mtab
        mount -f / || failed=1
        mount -f /proc || failed=1
        mount -f /sys || failed=1
        (exit ${failed})
        evaluate_retval

        # This will mount all filesystems that do not have _netdev in
        # their option list.  _netdev denotes a network filesystem.
        boot_mesg "Mounting remaining file systems..."
        mount -a -O no_netdev >/dev/null
        evaluate_retval
        ;;

    stop)
        boot_mesg "Unmounting all other currently mounted file systems..."
        umount -a -d -r >/dev/null
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        exit 1
        ;;
esac

# End $rc_base/init.d/mountfs
```

D.11. /etc/rc.d/init.d/udev_retry

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors     : Alexander E. Patrakov
```

```
#
# Version     : 00.02
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Retrying failed uevents, if any..."

        # From Debian: "copy the rules generated before / was mounted
        # read-write":
        for file in /dev/.udev/tmp-rules--*; do
            dest=${file##*tmp-rules--}
            [ "$dest" = '*' ] && break
            cat $file >> /etc/udev/rules.d/$dest
            rm -f $file
        done

        # Re-trigger the failed uevents in hope they will succeed now
        /sbin/udevadm trigger --retry-failed

        # Now wait for udevd to process the uevents we triggered
        /sbin/udevadm settle
        evaluate_retval
        ;;

    *)
        echo "Usage ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/udev_retry
```

D.12. /etc/rc.d/init.d/cleanfs

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/cleanfs
#
# Description : Clean file system
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
```

```
. ${rc_functions}

# Function to create files/directory on boot.
create_files() {
    # Read in the configuration file.
    exec 9>&0 < /etc/sysconfig/createfiles
        while read name type perm usr grp dtype maj min junk
        do

            # Ignore comments and blank lines.
                   case "${name}" in
                ""|\#*) continue ;;
            esac

            # Ignore existing files.
            if [ ! -e "${name}" ]; then
                # Create stuff based on its type.
                case "${type}" in
                    dir)
                        mkdir "${name}"
                        ;;
                    file)
                        :> "${name}"
                        ;;
                    dev)
                        case "${dtype}" in
                            char)
                                mknod "${name}" c ${maj} ${min}
                                ;;
                            block)
                                mknod "${name}" b ${maj} ${min}
                                ;;
                            pipe)
                                mknod "${name}" p
                                ;;
                            *)
                                boot_mesg -n "\nUnknown device type: ${dtype}" ${WARNING}
                                boot_mesg "" ${NORMAL}
                                ;;
                        esac
                        ;;
                    *)
                        boot_mesg -n "\nUnknown type: ${type}" ${WARNING}
                        boot_mesg "" ${NORMAL}
                        continue
                        ;;
                esac

                # Set up the permissions, too.
                chown ${usr}:${grp} "${name}"
                chmod ${perm} "${name}"
            fi
        done
    exec 0>&9 9>&-
}

case "${1}" in
```

```
    start)
        boot_mesg -n "Cleaning file systems:" ${INFO}

        boot_mesg -n " /tmp" ${NORMAL}
        cd /tmp &&
        find . -xdev -mindepth 1 ! -name lost+found \
            -delete || failed=1

        boot_mesg -n " /var/lock" ${NORMAL}
        cd /var/lock &&
        find . -type f -exec rm -f {} \; || failed=1

        boot_mesg " /var/run" ${NORMAL}
        cd /var/run &&
        find . ! -type d ! -name utmp \
            -exec rm -f {} \; || failed=1
        > /var/run/utmp
        if grep -q '^utmp:' /etc/group ; then
            chmod 664 /var/run/utmp
            chgrp utmp /var/run/utmp
        fi

        (exit ${failed})
        evaluate_retval

        if egrep -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
            boot_mesg "Creating files and directories..."
            create_files
            evaluate_retval
        fi
        ;;
    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/cleanfs
```

D.13. /etc/rc.d/init.d/console

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/console
#
# Description : Sets keymap and screen font
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#         Alexander E. Patrakov
#
# Version     : 00.03
#
# Notes       :
#
########################################################################

```

```
. /etc/sysconfig/rc
. ${rc_functions}

# Native English speakers probably don't have /etc/sysconfig/console at all
if [ -f /etc/sysconfig/console ]
then
    . /etc/sysconfig/console
else
        exit 0
fi

is_true() {
    [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

failed=0

case "${1}" in
    start)
        boot_mesg "Setting up Linux console..."
        # There should be no bogus failures below this line!

        # Figure out if a framebuffer console is used
        [ -d /sys/class/graphics/fb0 ] && USE_FB=1 || USE_FB=0

        # Figure out the command to set the console into the
        # desired mode
        is_true "${UNICODE}" &&
            MODE_COMMAND="${ECHO} -en '\033%G' && kbd_mode -u" ||
            MODE_COMMAND="${ECHO} -en '\033%@\033(K' && kbd_mode -a"

        # On framebuffer consoles, font has to be set for each vt in
        # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

        ! is_true "${USE_FB}" || [ -z "${FONT}" ] ||
            MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

        # Apply that command to all consoles mentioned in
        # /etc/inittab. Important: in the UTF-8 mode this should
        # happen before setfont, otherwise a kernel bug will
        # show up and the unicode map of the font will not be
        # used.
        # FIXME: Fedora Core also initializes two spare consoles
        # - do we want that?

        for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
            grep -o '\btty[[:digit:]]*\b'`
        do
            openvt -f -w -c ${TTY#tty} -- \
                /bin/sh -c "${MODE_COMMAND}" || failed=1
        done

        # Set the font (if not already set above) and the keymap
        is_true "${USE_FB}" || [ -z "${FONT}" ] ||
            setfont $FONT ||
            failed=1
        [ -z "${KEYMAP}" ] ||
```

```
            loadkeys ${KEYMAP} >/dev/null 2>&1 ||
            failed=1
        [ -z "${KEYMAP_CORRECTIONS}" ] ||
            loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
            failed=1

        # Convert the keymap from $LEGACY_CHARSET to UTF-8
        [ -z "$LEGACY_CHARSET" ] ||
            dumpkeys -c "$LEGACY_CHARSET" |
            loadkeys -u >/dev/null 2>&1 ||
            failed=1

        # If any of the commands above failed, the trap at the
        # top would set $failed to 1
        ( exit $failed )
        evaluate_retval
        ;;
    *)
        echo $"Usage:" "${0} {start}"
        exit 1
        ;;
esac

# End $rc_base/init.d/console
```

D.14. /etc/rc.d/init.d/localnet

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/localnet
#
# Description : Loopback device
#
# Authors     : Gerard Beekmans  - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. /etc/sysconfig/network

case "${1}" in
    start)
        boot_mesg "Bringing up the loopback interface..."
        ip addr add 127.0.0.1/8 label lo dev lo
        ip link set lo up
        evaluate_retval

        boot_mesg "Setting hostname to ${HOSTNAME}..."
        hostname ${HOSTNAME}
        evaluate_retval
        ;;
```

```
    stop)
        boot_mesg "Bringing down the loopback interface..."
        ip link set lo down
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        echo "Hostname is: $(hostname)"
        ip link show lo
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/localnet
```

D.15. /etc/rc.d/init.d/sysctl

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#               parameters
#
# Authors     : Nathan Coulson (nathan@linuxfromscratch.org)
#               Matthew Burgress (matthew@linuxfromscratch.org)
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        if [ -f "/etc/sysctl.conf" ]; then
            boot_mesg "Setting kernel runtime parameters..."
            sysctl -q -p
            evaluate_retval
        fi
        ;;
```

```
    status)
        sysctl -a
        ;;

    *)
        echo "Usage: ${0} {start|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/sysctl
```

D.16. /etc/rc.d/init.d/sysklogd

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/sysklogd
#
# Description : Sysklogd loader
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Starting system log daemon..."
        loadproc syslogd -m 0

        boot_mesg "Starting kernel log daemon..."
        loadproc klogd
        ;;

    stop)
        boot_mesg "Stopping kernel log daemon..."
        killproc klogd

        boot_mesg "Stopping system log daemon..."
        killproc syslogd
        ;;

    reload)
        boot_mesg "Reloading system log daemon config file..."
        reloadproc syslogd
        ;;

    restart)
        ${0} stop
        sleep 1
```

```
        ${0} start
        ;;

    status)
        statusproc syslogd
        statusproc klogd
        ;;

    *)
        echo "Usage: ${0} {start|stop|reload|restart|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/sysklogd
```

D.17. /etc/rc.d/init.d/network

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/network
#
# Description : Network Control Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#         Nathan Coulson - nathan@linuxfromscratch.org
#         Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. /etc/sysconfig/network

case "${1}" in
    start)
        # Start all network interfaces
        for file in ${network_devices}/ifconfig.*
        do
            interface=${file##*/ifconfig.}

            # skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]
            then
                continue
            fi

            IN_BOOT=1 ${network_devices}/ifup ${interface}
        done
        ;;

    stop)
```

```
        # Reverse list
        FILES=""
        for file in ${network_devices}/ifconfig.*
        do
            FILES="${file} ${FILES}"
        done

        # Stop all network interfaces
        for file in ${FILES}
        do
            interface=${file##*/ifconfig.}

            # skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]
            then
                continue
            fi

            IN_BOOT=1 ${network_devices}/ifdown ${interface}
        done
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart}"
        exit 1
        ;;
esac

# End /etc/rc.d/init.d/network
```

D.18. /etc/rc.d/init.d/sendsignals

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/sendsignals
#
# Description : Sendsignals Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
```

```
    stop)
        boot_mesg "Sending all processes the TERM signal..."
        killall5 -15
        error_value=${?}

        sleep ${KILLDELAY}

        if [ "${error_value}" = 0 ]; then
            echo_ok
        else
            echo_failure
        fi

        boot_mesg "Sending all processes the KILL signal..."
        killall5 -9
        error_value=${?}

        sleep ${KILLDELAY}

        if [ "${error_value}" = 0 ]; then
            echo_ok
        else
            echo_failure
        fi
        ;;

    *)
        echo "Usage: ${0} {stop}"
        exit 1
        ;;

esac

# End $rc_base/init.d/sendsignals
```

D.19. /etc/rc.d/init.d/reboot

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/reboot
#
# Description : Reboot Scripts
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    stop)
```

```
        boot_mesg "Restarting system..."
        reboot -d -f -i
        ;;

    *)
        echo "Usage: ${0} {stop}"
        exit 1
        ;;

esac

# End $rc_base/init.d/reboot
```

D.20. /etc/rc.d/init.d/halt

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/halt
#
# Description : Halt Script
#
# Authors     : Gerard Beekmans - gerard@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    stop)
        halt -d -f -i -p
        ;;
    *)
        echo "Usage: {stop}"
        exit 1
        ;;
esac

# End $rc_base/init.d/halt
```

D.21. /etc/rc.d/init.d/template

```
#!/bin/sh
########################################################################
# Begin $rc_base/init.d/
#
# Description :
#
# Authors     :
#
# Version     : 00.00
#
```

```
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

case "${1}" in
    start)
        boot_mesg "Starting..."
        loadproc
        ;;

    stop)
        boot_mesg "Stopping..."
        killproc
        ;;

    reload)
        boot_mesg "Reloading..."
        reloadproc
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        statusproc
        ;;

    *)
        echo "Usage: ${0} {start|stop|reload|restart|status}"
        exit 1
        ;;
esac

# End $rc_base/init.d/
```

D.22. /etc/sysconfig/rc

```
########################################################################
# Begin /etc/sysconfig/rc
#
# Description : rc script configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

```

```
rc_base=/etc/rc.d
rc_functions=${rc_base}/init.d/functions
network_devices=/etc/sysconfig/network-devices

# End /etc/sysconfig/rc
```

D.23. /etc/sysconfig/modules

```
########################################################################
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#          <module> [<arg1> <arg2> ...]
#
# Each module should be on it's own line, and any options that you want
# passed to the module should follow it.  The line deliminator is either
# a space or a tab.
########################################################################

# End /etc/sysconfig/modules
```

D.24. /etc/sysconfig/createfiles

```
########################################################################
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors     :
#
# Version     : 00.00
#
# Notes       : The syntax of this file is as follows:
#         if type is equal to "file" or "dir"
#          <filename> <type> <permissions> <user> <group>
#         if type is equal to "dev"
#          <filename> <type> <permissions> <user> <group> <devtype> <major> <minor>
#
#         <filename> is the name of the file which is to be created
#         <type> is either file, dir, or dev.
#               file creates a new file
#               dir creates a new directory
#               dev creates a new device
#         <devtype> is either block, char or pipe
#               block creates a block device
#               char creates a character deivce
#               pipe creates a pipe, this will ignore the <major> and <minor> fields
#         <major> and <minor> are the major and minor numbers used for the device.
########################################################################

```

```
# End /etc/sysconfig/createfiles
```

D.25. /etc/sysconfig/network-devices/ifup

```
#!/bin/sh
########################################################################
# Begin $network_devices/ifup
#
# Description : Interface Up
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the services directory, to indicate what file the
#               service should source to get environmental variables.
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Collect a list of configuration files for our interface
if [ -n "${2}" ]; then
    for file in ${@#$1} # All parameters except $1
  do
        FILES="${FILES} ${network_devices}/ifconfig.${1}/${file}"
    done
elif [ -d "${network_devices}/ifconfig.${1}" ]; then
    FILES=`echo ${network_devices}/ifconfig.${1}/*`
else
    FILES="${network_devices}/ifconfig.${1}"
fi

boot_mesg "Bringing up the ${1} interface..."
boot_mesg_flush

# Process each configruation file
for file in ${FILES}; do
    # skip backup files
    if [ "${file}" != "${file%""~""}" ]; then
        continue
    fi

    if [ ! -f "${file}" ]; then
        boot_mesg "${file} is not a network configuration file or directory." ${WARNING}
        echo_warning
        continue
    fi

    (
        . ${file}

        # Will not process this service if started by boot, and ONBOOT
```

```
        # is not set to yes
        if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
            continue
        fi
        # Will not process this service if started by hotplug, and
        # ONHOTPLUG is not set to yes
        if [ "${IN_HOTPLUG}" = "1" -a "${ONHOTPLUG}" != "yes" \
                -a "${HOSTNAME}" != "(none)" ]; then continue
        fi

        if [ -n "${SERVICE}" -a -x "${network_devices}/services/${SERVICE}" ]; then
            if [ -z "${CHECK_LINK}" -o "${CHECK_LINK}" = "y" \
                    -o "${CHECK_LINK}" = "yes" -o "${CHECK_LINK}" = "1" ]; then
                if ip link show ${1} > /dev/null 2>&1; then
                    link_status=`ip link show ${1}`
                    if [ -n "${link_status}" ]; then
                        if ! echo "${link_status}" | grep -q UP; then
                            ip link set ${1} up
                        fi
                    fi
                else
                    boot_mesg "Interface ${1} doesn't exist." ${WARNING}
                    echo_warning
                    continue
                fi
            fi
            IFCONFIG=${file} ${network_devices}/services/${SERVICE} ${1} up
        else
            boot_mesg "Unable to process ${file}.  Either" ${FAILURE}
            boot_mesg " the SERVICE variable was not set,"
            boot_mesg " or the specified service cannot be executed."
            echo_failure
            continue
        fi
    )
done

# End $network_devices/ifup
```

D.26. /etc/sysconfig/network-devices/ifdown

```
#!/bin/sh
########################################################################
# Begin $network_devices/ifdown
#
# Description : Interface Down
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.01
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the services directory, to indicate what file the
#               service should source to get environmental variables.
#
```

```
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}

# Collect a list of configuration files for our interface
if [ -n "${2}" ]; then
    for file in ${@#$1}; do # All parameters except $1
        FILES="${FILES} ${network_devices}/ifconfig.${1}/${file}"
    done
elif [ -d "${network_devices}/ifconfig.${1}" ]; then
    FILES=`echo ${network_devices}/ifconfig.${1}/*`
else
    FILES="${network_devices}/ifconfig.${1}"
fi

# Reverse the order configuration files are processed in
for file in ${FILES}; do
    FILES2="${file} ${FILES2}"
done
FILES=${FILES2}

# Process each configuration file
for file in ${FILES}; do
    # skip backup files
    if [ "${file}" != "${file%""~""}" ]; then
        continue
    fi

    if [ ! -f "${file}" ]; then
        boot_mesg "${file} is not a network configuration file or directory." ${WARNING}
        echo_warning
        continue
    fi
    (
        . ${file}

        # Will not process this service if started by boot, and ONBOOT
        # is not set to yes
        if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
            continue
        fi

        # Will not process this service if started by hotplug, and
        # ONHOTPLUG is not set to yes
        if [ "${IN_HOTPLUG}" = "1" -a "${ONHOTPLUG}" != "yes" ]; then
            continue
        fi

        # This will run the service script, if SERVICE is set
        if [ -n "${SERVICE}" -a -x "${network_devices}/services/${SERVICE}" ]; then
            if ip link show ${1} > /dev/null 2>&1
            then
                IFCONFIG=${file} ${network_devices}/services/${SERVICE} ${1} down
            else
                boot_mesg "Interface ${1} doesn't exist." ${WARNING}
                echo_warning
```

```
            fi
        else
            boot_mesg -n "Unable to process ${file}.  Either" ${FAILURE}
            boot_mesg -n " the SERVICE variable was not set,"
            boot_mesg " or the specified service cannot be executed."
            echo_failure
            continue
        fi
    )
done

if [ -z "${2}" ]; then
    link_status=`ip link show $1`
    if [ -n "${link_status}" ]; then
        if echo "${link_status}" | grep -q UP; then
            boot_mesg "Bringing down the ${1} interface..."
            ip link set ${1} down
            evaluate_retval
        fi
    fi
fi

# End $network_devices/ifdown
```

D.27. /etc/sysconfig/network-devices/services/ipv4-static

```
#!/bin/sh
########################################################################
# Begin $network_devices/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors     : Nathan Coulson - nathan@linuxfromscratch.org
#         Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. ${IFCONFIG}

if [ -z "${IP}" ]; then
    boot_mesg "IP variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
    echo_failure
    exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
    boot_mesg -n "PREFIX variable missing from ${IFCONFIG}," ${WARNING}
    boot_mesg " assuming 24."
    echo_warning
    PREFIX=24
```

```
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
    boot_mesg "PREFIX and PEER both specified in ${IFCONFIG}, cannot continue." ${FAILURE}
    echo_failure
    exit 1
elif [ -n "${PREFIX}" ]; then
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PEER}" ]; then
    args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${BROADCAST}" ]; then
    args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
    up)
        boot_mesg "Adding IPv4 address ${IP} to the ${1} interface..."
        ip addr add ${args} dev ${1}
        evaluate_retval

        if [ -n "${GATEWAY}" ]; then
            if ip route | grep -q default; then
                boot_mesg "Gateway already setup; skipping." ${WARNING}
                echo_warning
            else
                boot_mesg "Setting up default gateway..."
                ip route add default via ${GATEWAY} dev ${1}
                evaluate_retval
             fi
        fi
    ;;

    down)
        if [ -n "${GATEWAY}" ];    then
            boot_mesg "Removing default gateway..."
            ip route del default
            evaluate_retval
        fi

        boot_mesg "Removing IPv4 address ${IP} from the ${1} interface..."
        ip addr del ${args} dev ${1}
        evaluate_retval
    ;;

    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1
    ;;
esac

# End $network_devices/services/ipv4-static
```

D.28. /etc/sysconfig/network-devices/services/ipv4-static-route

```
#!/bin/sh
```

```
########################################################################
# Begin $network_devices/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors     : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#
# Version     : 00.00
#
# Notes       :
#
########################################################################

. /etc/sysconfig/rc
. ${rc_functions}
. ${IFCONFIG}

case "${TYPE}" in
    ("" | "network")
        need_ip=1
        need_gateway=1
    ;;

    ("default")
        need_gateway=1
        args="${args} default"
        desc="default"
    ;;

    ("host")
        need_ip=1
    ;;

    ("unreachable")
        need_ip=1
        args="${args} unreachable"
        desc="unreachable "
    ;;

    (*)
        boot_mesg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
    ;;
esac

if [ -n "${need_ip}" ]; then
    if [ -z "${IP}" ]; then
        boot_mesg "IP variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
    fi

    if [ -z "${PREFIX}" ]; then
        boot_mesg "PREFIX variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
```

```
    fi

    args="${args} ${IP}/${PREFIX}"
    desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
    if [ -z "${GATEWAY}" ]; then
        boot_mesg "GATEWAY variable missing from ${IFCONFIG}, cannot continue." ${FAILURE}
        echo_failure
        exit 1
    fi
    args="${args} via ${GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
        args="${args} src ${SOURCE}"
fi

case "${2}" in
    up)
        boot_mesg "Adding '${desc}' route to the ${1} interface..."
        ip route add ${args} dev ${1}
        evaluate_retval
    ;;

    down)
        boot_mesg "Removing '${desc}' route from the ${1} interface..."
        ip route del ${args} dev ${1}
        evaluate_retval
    ;;

    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1
    ;;
esac

# End $network_devices/services/ipv4-static-route
```

Appendix E. Udev configuration rules

The rules from udev-config-20081015.tar.bz2 in this appendix are listed for convenience. Installation is normally done via instructions in Section 6.56, “Udev-130”.

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# override both of these
KERNEL=="random",   MODE="0444"
KERNEL=="urandom",  MODE="0444"

KERNEL=="aio",      MODE="0444"
KERNEL=="kmsg",     MODE="0600"
KERNEL=="rtc",      MODE="0666"

# Comms devices

KERNEL=="rfcomm[0-9]*",     GROUP="uucp"
KERNEL=="ippp[0-9]*",       GROUP="uucp"
KERNEL=="isdn[0-9]*",       GROUP="uucp"
KERNEL=="isdnctrl[0-9]*",   GROUP="uucp"
KERNEL=="capi",             NAME="capi20",  SYMLINK+="isdn/capi20"
KERNEL=="capi?*",           NAME="capi/%n", GROUP="uucp"
KERNEL=="dcbri[0-9]*",      GROUP="uucp"

# ALSA devices go in their own subdirectory

KERNEL=="controlC[0-9]*",        GROUP="audio",  NAME="snd/%k"
KERNEL=="hwC[0-9]*D[0-9]*",      GROUP="audio",  NAME="snd/%k"
KERNEL=="pcmC[0-9]*D[0-9]*[cp]", GROUP="audio",  NAME="snd/%k"
KERNEL=="midiC[0-9]*D[0-9]*",    GROUP="audio",  NAME="snd/%k"
KERNEL=="timer",                 GROUP="audio",  NAME="snd/%k"
KERNEL=="seq",                   GROUP="audio",  NAME="snd/%k"

# Sound devices

KERNEL=="admmidi*",     GROUP="audio"
KERNEL=="adsp*",        GROUP="audio"
KERNEL=="aload*",       GROUP="audio"
KERNEL=="amidi*",       GROUP="audio"
KERNEL=="amixer*",      GROUP="audio"
KERNEL=="audio*",       GROUP="audio"
KERNEL=="dmfm*",        GROUP="audio"
KERNEL=="dmmidi*",      GROUP="audio"
KERNEL=="dsp*",         GROUP="audio"
KERNEL=="midi*",        GROUP="audio"
KERNEL=="mixer*",       GROUP="audio"
KERNEL=="music",        GROUP="audio"
KERNEL=="sequencer*",   GROUP="audio"

# Input devices
```

```
# override MODE on these four
KERNEL=="mice",     MODE="0644",    SYMLINK+="mouse"
KERNEL=="mouse*",   MODE="0644"
KERNEL=="event*",   MODE="0644"
KERNEL=="ts*",      MODE="0644"

KERNEL=="psaux",    MODE="0644"
KERNEL=="js",       MODE="0644"
KERNEL=="djs",      MODE="0644"

# USB devices go in their own subdirectory

KERNEL=="hiddev*",          NAME="usb/%k"
KERNEL=="legousbtower*",    NAME="usb/%k"
KERNEL=="dabusb*",          NAME="usb/%k"
SUBSYSTEMS=="usb", KERNEL=="lp[0-9]*", NAME="usb/%k"

# DRI devices are managed by the X server, so prevent udev from creating them

KERNEL=="card*",    OPTIONS+="ignore_device"

# Video devices

KERNEL=="fb[0-9]*",     GROUP="video"
KERNEL=="video[0-9]*",  GROUP="video"
KERNEL=="radio[0-9]*",  GROUP="video"
KERNEL=="vbi[0-9]*",    GROUP="video"
KERNEL=="vtx[0-9]*",    GROUP="video"

# DVB devices

SUBSYSTEM=="dvb", GROUP="video"

# Storage/memory devices

# override: make group-writable
SUBSYSTEM=="block", MODE="0660"

# dmsetup and lvm2 related programs create devicemapper devices so we prevent
# udev from creating them

KERNEL=="dm-*",             OPTIONS+="ignore_device"

# Tape devices

# override all these
KERNEL=="ht[0-9]*",     GROUP="tape"
KERNEL=="nht[0-9]*",    GROUP="tape"
KERNEL=="pt[0-9]*",     GROUP="tape"
KERNEL=="npt[0-9]*",    GROUP="tape"
KERNEL=="st[0-9]*",     GROUP="tape"
KERNEL=="nst[0-9]*",    GROUP="tape"

# Override floppy devices
KERNEL=="fd[0-9]",    GROUP="floppy"
KERNEL=="fd[0-9]", ACTION=="add", ATTRS{cmos}=="?*", \
```

```
  RUN+="create_floppy_devices -c -t $attr{cmos} -m %M -M 0660 -G floppy $root/%k"
```

E.2. 61-cdrom.rules

```
# /etc/udev/rules.d/61-cdrom.rules: Set CD-ROM permissions.

ACTION=="add", SUBSYSTEM=="block", ENV{ID_TYPE}=="cd", GROUP="cdrom"
```

Appendix F. LFS Licenses

This book is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 2.0 License.

Computer instructions may be extracted from the book under the MIT License.

F.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Important

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions
   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.
   d. "Original Author" means the individual or entity who created the Work.
   e. "Work" means the copyrightable work of authorship offered under the terms of this License.
   f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

   g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.
3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:
   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
   b. to create and reproduce Derivative Works;
   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;
   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).
4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:
   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.
   b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform,

or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Important

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

F.2. The MIT License

Copyright © 1999-2008 Gerard Beekmans

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Index

Packages

Programs

Libraries

Scripts

Others