

# Linux From Scratch ( 简体中文版 )

Version 8.4-systemd

项目创建者: Gerard Beekmans

总编: Bruce Dubbs

编辑: DJ Lucas

翻译: Linux 中国 - LCTT - LFS 翻译小组

# Linux From Scratch ( 简体中文版 ) : Version 8.4-systemd

by 项目创建者 : Gerard Beekmans, 总编 : Bruce Dubbs, 编辑 : DJ Lucas, and 翻译 : Linux 中国 - LCTT - LFS 翻译小组

Copyright © 1999-2019 Gerard Beekmans

Copyright © 1999-2019, Gerard Beekmans

版权所有。

本手册根据 Creative Commons License 授权。

手册中的计算机指令可根据 MIT License 摘录。

Linux® 为 Linus Torvalds 注册商标。

# Table of Contents

序章 .....	vii
i. 前言 .....	vii
ii. 致读者 .....	vii
iii. LFS 的目标架构 .....	viii
iv. LFS 和标准 .....	viii
v. 本书中的软件包逻辑 .....	ix
vi. 前提条件 .....	xiii
vii. 排版约定 .....	xiii
viii. 本书结构 .....	xiv
ix. 勘误表 .....	xiv
I. 介绍 .....	1
1. 介绍 .....	2
1.1. 如何构建 LFS 系统 .....	2
1.2. What's new since the last release .....	2
1.3. Changelog .....	3
1.4. 资源 .....	6
1.5. 帮助 .....	7
II. 准备构建 .....	9
2. 准备宿主系统 .....	10
2.1. 简介 .....	10
2.2. Host System Requirements .....	10
2.3. 分阶段构建 LFS .....	13
2.4. 创建新分区 .....	13
2.5. 在分区上创建文件系统 .....	14
2.6. 设置 \$LFS 变量 .....	15
2.7. 挂载新分区 .....	15
3. 软件包和补丁 .....	17
3.1. 简介 .....	17
3.2. All Packages .....	17
3.3. Needed Patches .....	23
4. 最后的准备工作 .....	25
4.1. 简介 .....	25
4.2. 创建目录 \$LFS/tools .....	25
4.3. 添加 LFS 用户 .....	25
4.4. 设置环境 .....	26
4.5. About SBUs .....	26
4.6. 关于测试套件 .....	27
5. Constructing a Temporary System .....	28
5.1. 简介 .....	28
5.2. 工具链技术说明 .....	28
5.3. 通用编译指南 .....	29
5.4. Binutils-2.32 - 第 1 遍 .....	31
5.5. GCC-8.2.0 - 第 1 遍 .....	33
5.6. Linux-4.20.12 API 头文件 .....	35
5.7. Glibc-2.29 .....	36
5.8. GCC-8.2.0 中的 Libstdc++ .....	38
5.9. Binutils-2.32 - 第 2 遍 .....	39
5.10. GCC-8.2.0 - 第 2 遍 .....	40
5.11. Tcl-8.6.9 .....	42
5.12. Expect-5.45.4 .....	43
5.13. DejaGNU-1.6.2 .....	44
5.14. M4-1.4.18 .....	45
5.15. Ncurses-6.1 .....	46
5.16. Bash-5.0 .....	47
5.17. Bison-3.3.2 .....	48
5.18. Bzip2-1.0.6 .....	49
5.19. Coreutils-8.30 .....	50
5.20. Diffutils-3.7 .....	51

5.21. File-5.36 .....	52
5.22. Findutils-4.6.0 .....	53
5.23. Gawk-4.2.1 .....	54
5.24. Gettext-0.19.8.1 .....	55
5.25. Grep-3.3 .....	56
5.26. Gzip-1.10 .....	57
5.27. Make-4.2.1 .....	58
5.28. Patch-2.7.6 .....	59
5.29. Perl-5.28.1 .....	60
5.30. Python-3.7.2 .....	61
5.31. Sed-4.7 .....	62
5.32. Tar-1.31 .....	63
5.33. Texinfo-6.5 .....	64
5.34. Util-linux-2.33.1 .....	65
5.35. Xz-5.2.4 .....	66
5.36. 清理无用内容 .....	67
5.37. 改变属主 .....	67
III. 构建 LFS 系统 .....	68
6. Installing Basic System Software .....	69
6.1. 简介 .....	69
6.2. 准备虚拟内核文件系统 .....	69
6.3. 软件包管理 .....	70
6.4. 进入 Chroot 环境 .....	72
6.5. 创建目录 .....	73
6.6. Creating Essential Files and Symlinks .....	73
6.7. Linux-4.20.12 API 头文件 .....	77
6.8. Man-pages-4.16 .....	78
6.9. Glibc-2.29 .....	79
6.10. 调整工具链 .....	85
6.11. Zlib-1.2.11 .....	87
6.12. File-5.36 .....	88
6.13. Readline-8.0 .....	89
6.14. M4-1.4.18 .....	90
6.15. Bc-1.07.1 .....	91
6.16. Binutils-2.32 .....	92
6.17. GMP-6.1.2 .....	94
6.18. MPFR-4.0.2 .....	96
6.19. MPC-1.1.0 .....	97
6.20. Shadow-4.6 .....	98
6.21. GCC-8.2.0 .....	101
6.22. Bzip2-1.0.6 .....	105
6.23. Pkg-config-0.29.2 .....	107
6.24. Ncurses-6.1 .....	108
6.25. Attr-2.4.48 .....	110
6.26. Acl-2.2.53 .....	111
6.27. Libcap-2.26 .....	112
6.28. Sed-4.7 .....	113
6.29. Psmisc-23.2 .....	114
6.30. Iana-Etc-2.30 .....	115
6.31. Bison-3.3.2 .....	116
6.32. Flex-2.6.4 .....	117
6.33. Grep-3.3 .....	118
6.34. Bash-5.0 .....	119
6.35. Libtool-2.4.6 .....	120
6.36. GDBM-1.18.1 .....	121
6.37. Gperf-3.1 .....	122
6.38. Expat-2.2.6 .....	123
6.39. Inetutils-1.9.4 .....	124
6.40. Perl-5.28.1 .....	126
6.41. XML::Parser-2.44 .....	128
6.42. Intltool-0.51.0 .....	129

6.43. Autoconf-2.69 .....	130
6.44. Automake-1.16.1 .....	131
6.45. Xz-5.2.4 .....	132
6.46. Kmod-26 .....	134
6.47. Gettext-0.19.8.1 .....	135
6.48. Libelf from Elfutils-0.176 .....	137
6.49. Libffi-3.2.1 .....	138
6.50. OpenSSL-1.1.1a .....	139
6.51. Python-3.7.2 .....	140
6.52. Ninja-1.9.0 .....	142
6.53. Meson-0.49.2 .....	143
6.54. Coreutils-8.30 .....	144
6.55. Check-0.12.0 .....	148
6.56. Diffutils-3.7 .....	149
6.57. Gawk-4.2.1 .....	150
6.58. Findutils-4.6.0 .....	151
6.59. Groff-1.22.4 .....	152
6.60. GRUB-2.02 .....	154
6.61. Less-530 .....	156
6.62. Gzip-1.10 .....	157
6.63. IPRoute2-4.20.0 .....	158
6.64. Kbd-2.0.4 .....	160
6.65. Libpipeline-1.5.1 .....	162
6.66. Make-4.2.1 .....	163
6.67. Patch-2.7.6 .....	164
6.68. Man-DB-2.8.5 .....	165
6.69. Tar-1.31 .....	167
6.70. Texinfo-6.5 .....	168
6.71. Vim-8.1 .....	170
6.72. systemd-240 .....	173
6.73. D-Bus-1.12.12 .....	177
6.74. Procs-ng-3.3.15 .....	179
6.75. Util-linux-2.33.1 .....	181
6.76. E2fsprogs-1.44.5 .....	185
6.77. 关于调试符号 .....	188
6.78. 再次清理无用内容 .....	188
6.79. 清理 .....	189
7. 基本系统配置 .....	190
7.1. 简介 .....	190
7.2. 通用网络配置 .....	190
7.3. 设备与模块管理概述 .....	193
7.4. 设备管理 .....	195
7.5. 配置系统时间 .....	196
7.6. 配置 Linux 控制台 .....	196
7.7. 配置系统语言环境 .....	197
7.8. Creating the /etc/inputrc File .....	199
7.9. Creating the /etc/shells File .....	199
7.10. systemd Usage and Configuration .....	200
8. 让 LFS 系统可引导 .....	203
8.1. 简介 .....	203
8.2. 创建 /etc/fstab 文件 .....	203
8.3. Linux-4.20.12 .....	204
8.4. Using GRUB to Set Up the Boot Process .....	208
9. 尾声 .....	210
9.1. 最后的最后 .....	210
9.2. 为 LFS 用户数添砖加瓦 .....	210
9.3. 重启系统 .....	210
9.4. 接下来做什么呢? .....	211
IV. 附录 .....	213
A. 缩写和术语 .....	214
B. Acknowledgments .....	216

C. Dependencies .....	218
D. LFS Licenses .....	228
D.1. Creative Commons License .....	228
D.2. The MIT License .....	231
Index .....	232

# 序章

## 前言

自 1998 始，我逐渐从学习 Linux 到对其深入了解。仅仅是在我安装完我的第一个 Linux 发行版后，我便迅速地 Linux 所蕴含的整套理念及其背后的哲学思想所着迷。

总是会有很多的方法来完成一个任务。这对 Linux 的发行版而言也是这样。这些年来，这么多的发行版层出不穷，有的还在，有的已经今非昔比，有的现在也只能在记忆中感怀。它们的行事风格大相径庭，以满足不同受众的需求。也正是因为如此之多不同的发行版想要到达的是同一个目标，我开始意识到自己可以不必在受限于别人做好的发行版。在开始探索 Linux 之前，我们只能忍受各种操作系统的问题，因为我们别无选择。无论你喜欢还是不喜欢，它就在那里。但 Linux 让你拥有了选择的可能。如果你有什么不喜欢的，你可以自由地，甚至倍受鼓舞地去改变它。

我尝试过很多发行版，但还是无法决定该用哪个。这些系统，从它们自身的角度来看，它们都是很棒的系统。这里也并不是在探讨对错问题，这只不过是个人习惯的问题。在各种可选项中，显然没有哪个单一的系统对我而言是完美的。于是我开始着手创建自己的 Linux 系统，以完全满足自己的喜好。

为了使系统真正地属于我，我毅然决定开始从源代码编译所有东西，而不是使用预编译好的二进制包。这个「完美」的 Linux 系统能够拥有不同系统的优点，而不会有它们与之俱来的不足。起初，这种想法让人想都不敢想，但我一直保持着可以构建出这样一个系统的信念。

在梳理了诸如循环依赖和编译错误之类的问题后，我最终构建出了一个订制的 Linux 系统。它运转自如，用起来与当时的那些 Linux 系统一样完美，但它是由我亲手打造的。能组装出这样一个系统让我很满意，而与之相比更棒的，恐怕唯有在软件的各个方面都亲历亲为了，不过这也可算是第二棒的事了。

当我将自己的目标和经验与 Linux 社区的其它成员分享时，引发了大家对这些想法的持续关注。很快人们就清楚地意识到，这样一个定制的 Linux 系统不仅仅可以满足用户的特定需求，还可为程序员和系统管理员提供一个理想的学习机会，以增强他们（原有）的 Linux 技能。出于这个被放大的兴趣，Linux From Scratch 项目诞生了。

这本 Linux From Scratch 手册是该项目的重中之重。它为你提供了设计和构建自己的系统所需的背景知识和指令。虽然本手册给出了一个模板，一个最后能正常工作的系统，但你可以自由地改变指令以适应自身的需求，从某种程度上说，这是也本项目的一个重要部分。你仍然可以控制一切，我们只是在你起航之前助你一臂之力。

我衷心地希望你能享受构建属于自己的 Linux From Scratch 系统的过程，并对坐拥一个真正属于自己的系统所带来的诸多好处能做到甘之如饴。

--  
Gerard Beekmans  
gerard@linuxfromscratch.org

## 致读者

说到为什么要读这本书，我想，原因一定有很多。有些人可能会提出这样的疑问：「都已经现成的 Linux 系统可以下载和安装了，你为什么还要多此一举的从头构建一个呢？」

此项目存在的一个重要原因是帮助你了解 Linux 的内部是如何工作的。通过构建 LFS 系统，你可以更好地理解 Linux 的工作原理，以及程序是如何协同工作和相互依赖的。最棒的是，这个学习经历能给你提供自定义 Linux 系统以满足你自己独特需求的能力。

另一个重要的原因是，你对系统本身有更多的控制权，而无需去关心别人的 Linux 是如何实现的。在 LFS 下，你就是主宰，系统的各个方面都需要你亲力亲为。

LFS 可以让你创建极其精简的 Linux 系统。在安装那些常规的 Linux 系统时，你往往会迫不得已而安装那些你用不到的（甚至你都不知道它们是干什么用的）程序。这些程序也许会浪费你的硬件资源。或许你可能会说，现在计算机的资源那么丰富，稍微浪费一些又有什么关系呢。但是你依旧要考虑到可引导 CD、U 盘或者是一些嵌入式的环境，它们对资源高度敏感，这也恰恰是 LFS 所擅长的地方。

另一个优势便是，自定义的 Linux 系统有着更高的安全性。通过从源码构建一个完整的系统，你有权审核所有的代码以及所打入的安全补丁。也就没必要去在等别人编译修复该安全漏洞的二进制程序了。而且，除非你亲自检查了补丁文件并且做了完整的测试，否则你又怎么能确信，新的二进制程序确实是正确的编译且解决了问题呢？

从头构建一个基本可用的 Linux 系统是本书的目标，你即使不打算这么做，也还是能从这本书的内容中有所收获。

支撑你从头构建属于自己的 LFS 系统的原因实在是太多了。但说到最后，其中最重要的一个原因还是教学。随着你的 LFS 经验愈趋熟稔，这些信息和知识真正给你带来的力量，你也便了然于胸了。

## LFS 的目标架构

LFS 当前主要支持 AMD/Intel 的 x86 (32 位) 和 x86\_64 (64 位) 构架的 CPU。另外，本文档也涉及一些更改以允许 LFS 顺利地在 Power PC 和 ARM CPU 上运行。为了顺利构建 LFS，除了后面几页的内容外，你主要需要一个可以在当前 CPU 上正常运行的 Linux 系统，例如：早先版本的 LFS、Ubuntu、Red Hat/Fedora、SuSE 或者是在你的架构上可以运行的其它发行版。另外注意 32 位的发行版是可以在 64 位的 AMD/Intel 处理器上作为宿主机正常安装和运作的。

这里也说明一下关于 64 位系统的一些情况。与 32 位相比，大体上程序运行的速度虽稍微地快了那么一点点，但体积也稍微的大了那么一点点。以在 Core 2 Duo 处理器上运行的 LFS-6.5 系统为例，以下便是实测数据：

Architecture	Build Time	Build Size
32-bit	198.5 minutes	648 MB
64-bit	190.6 minutes	709 MB

正如你看到的，64 位程序仅仅比 32 位程序快了 4%，而体积大了 9%。由此可见，单纯的追逐 64 位系统其实并没有太大的必要。但是，假如你的电脑的内存超过了 4 GB 又或者说需要操作大于 4 GB 的数据，64 位系统的优势就比较明显了。

### Note

上述讨论只适用于对比相同硬件下的构建过程。现代的 64 位系统会比老的 64 位系统更快。LFS 作者建议，如果有机会，尽量在 64 位系统上进行构建。

假如按照本文的默认方式构建，那么你将得到一个「纯」64 位系统——这意味着你仅能够执行 64 位的程序。构建「multi-lib」并不是不可以，但是这意味着很多的程序都需要编译两次：一次为 32 位程序编译，一次为 64 位程序编译。不过，本文档并不涉及这部份的内容，因为这些内容会干扰用户学习如何构建一份最基本的 Linux 系统。你可以通过阅读 Cross Linux From Scratch 的相关内容获得有关该话题的更多帮助。

## LFS 和标准

LFS 的结构尽可能的遵循 Linux 的标准。主要的标准有：

- POSIX.1-2008.
- 文件系统层次结构标准 (FHS) 3.0 版本
- Linux 标准规范 (LSB) 5.0 (2015) 版本

LSB 有四个独立的标准：核心 (Core)、桌面 (Desktop)、运行时语言 (Runtime Languages) 和成像 (Imaging)。除了通用要求，还有架构特定要求。此外还有两个领域在试行：分别是 Gtk3 和图形 (Graphics)。LFS 试图遵从前一节中所讨论的架构要求。

### Note

很多人不认同 LSB 的要求。定义它的主要目的是确保私有软件能够在兼容的系统上安装并正常运行。由于 LFS 是基于源代码的，用户对于需要什么软件包有完全的控制权，有很多人就选择不安装 LSB 规范要求的软件包。

完全可以创建一个能够通过 LSB 认证测试的完整 LFS 系统，但这需要很多 LFS 范围之外的额外软件包。在 BLFS 中有这些额外软件包的安装说明。

## 由 LFS 提供，用于满足 LSB 要求的软件包

LSB 核心：	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib
LSB 桌面：	无
LSB 运行时语言：	Perl



LSB 成像：无  
 LSB Gtk3 和 LSB 图形（试用）：无

## 由 BLFS 提供，用于满足 LSB 要求的软件包

LSB 核心：At, Batch (At 的一部分), Cpio, Ed, Fcron, Initd-tools, Lsb\_release, NSPR, NSS, PAM, Pax, Sendmail (或 Postfix 或 Exim), time  
 LSB 桌面：Alda, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdtk-pixbuf, Glib2, GTK+2, Icon-naming-utils, Libjpeg-turbo, Libpng, Libtiff, Libxml2, MesaLib, Pango, Xdg-utils, Xorg  
 LSB 运行时语言：Python, Libxml2, Libxslt  
 LSB 成像：CUPS, Cups-filters, Ghostscript, SANE  
 LSB Gtk3 和 LSB 图形（试用）：GTK+3

## LFS 和 BLFS 没有提供，用于满足 LSB 要求的软件包

LSB 核心：无  
 LSB 桌面：Qt4 (以及 Qt5 除外)  
 LSB 运行时语言：无  
 LSB 成像：无  
 LSB Gtk3 和 LSB 图形（试用）：无

## 本书中的软件包逻辑

正如前文所述，LFS 的目标是构建一个完整可用的基础级系统。这其中便包含了，所有地软件包需要复制自身，直至为用户提供出一个相对最小的基础，然后根据用户的选择，在这个基础上定制出一个更完善的系统。这便意味着 LFS 并不是最小可用系统的代名词。其中包含的几个重要软件包，也并非严格需求的。下面的列表介绍了本书中选择每个软件包的理由。

- Acl

这个软件包中，包含管理访问控制列表（ACL）的工具，用于定义文件和目录更细粒度的自主访问权。

- Attr

这个软件包中包含的程序，用于管理文件系统对象的扩展属性。

- Autoconf

这个软件包中包含的程序，能根据开发者的模板自动生成配置源代码的 shell 脚本。通常需要在更新构建过程之后重新构建一个软件包。

- Automake

这个软件包中，包含了通过模板生成 Make 文件的程序。通常需要在更新构建过程之后重新构建一个软件包。

- Bash

这个软件包用于能满足 LSB 核心的需求，为系统提供 Bourne Shell 接口。选择它而非其它 shell 软件包的理由，是因为它的通用性以及基本 shell 功能上的扩展能力。

- Bc

这个软件包提供了一种任意精度的数值处理语言。在构建 Linux 内核时需要它。

- Binutils

这个软件包中，包含了一个链接器、一个汇编器和其它处理对象文件的工具。编译 LFS 以及 BLFS 系统中的大部分软件包，包需要该软件包中的程序。

- Bison

这个软件包中有 GNU 版的 yacc (Yet Another Compiler Compiler)，用于构建一些其它的 LFS 程序。

- Bzip2

这个软件包中，包含用来压缩和解压缩文件的程序。在解压缩很多 LFS 软件包的时候需要它。

- Check

这个软件包中，包含一个用于测试其它程序的工具。仅安装在临时工具链中。

- Coreutils

这个软件包，包含了一些查看和管理文件及目录的重要程序。在命令行里管理文件时和每个 LFS 软件包的安装过程中需要它。

- D-Bus

这个软件包中，包含了一个用于提供消息总线的程序，消息总线是一种应用程序间通信的简单方式。

- DejaGNU

这个软件包中，包含一个测试其它程序的框架。仅安装在临时工具链中。

- Diffutils

这个软件包中，包含了一些显示文件和目录差异的程序。这些程序可以用来创建补丁，也用于很多软件包的构建过程。

- E2fsprogs

这个软件包中，包含了一些处理 ext2、ext3 和 ext4 文件系统的工具。这些是 Linux 上支持的最常用而且完全经过考验的文件系统。

- Expat

这个软件包中，包含了一个相对小的 XML 解析库。Perl 模块 XML::Parser 需求该软件包。

- Expect

这个软件包中，包含了一个生成与其它程序交互的脚本对话框的程序。通常用来测试其它软件包。仅安装在临时工具链中。

- File

这个软件包中，包含一个能判断给定文件的类型的工具。一些软件包需要用它来构建。

- Findutils

这个软件包中，包含了一些在文件系统中查找文件的程序。在很多软件包构建脚本中会用到。

- Flex

这个软件包中，包含了一个能生成识别文本模式程序的工具。是 lex (lexical analyzer) 程序的 GNU 版本。构建很多 LFS 软件包需要用到。

- Gawk

这个软件包中，包含了一些操作文本文件的程序。是 awk (Aho-Weinberg-Kernighan) 的 GNU 版本。在很多软件包的构建脚本中会用到。

- Gcc

这个软件包是 GNU 编译器工具集。它包括 C 和 C++ 的编译器以及其它构建 LFS 时不用到的软件包。

- GDBM

这个软件包中，包含 GNU 数据库管理库。LFS 的另一个软件包 Man-DB 会用到。

- Gettext

这个软件包中，包含了很多软件包国际化和本地化需要用到的工具和库。

- Glibc

这个软件包中，包含了主要的 C 语言库。缺少它 Linux 的程序便无法运行了。

- GMP

这个软件包中，包含了能提供任意精度数值运算的数学库。编译 Gcc 时会用到。

- Gperf

这个软件包中，包含了一个能从一个键集生成完美哈希函数的程序。Eudev 会用到。

- Grep

这个软件包中，包含了一些在文件中搜索的程序。大部分软件包的构建脚本会用到。

- Groff

这个软件包中，包含了处理和格式化文本的程序。其中一个重要的功能是格式化 man 页面。

- GRUB

这个软件包是 Grand Unified Boot Loader。是诸多可用的引导加载器中的一个，但是最灵活。

- Gzip

这个软件包中，包含了一些压缩和解压缩文件的程序。解压很多 LFS 以及 BLFS 的软件包时需要用到。

- Iana-etc

这个软件包中，包含提供了网络服务和协议的数据。启用合适的网络功能时会用到。

- Inetutils

这个软件包中，包含基本网络管理的程序。

- Intltool

这个软件包中，包含能从源文件中抽取可翻译字符串的工具。

- IProute2

这个软件包中，包含了一些基本和高级的 IPv4 和 IPv6 网络的程序。由于它的 IPv6 功能，所以选择它来取代其它的常见网络工具包 (net-tools)。

- Kbd

这个软件包中，包含一些键盘映射文件，用于非 US 键盘的键盘工具以及一些控制台字体。

- Kmod

这个软件包中，包含一些用于管理 Linux 内核模块的程序。

- Less

这个软件包中，包含了一个很好的文本文件查看器，允许查看文件的时候向上或向下滚动。Man-DB 用它来查看 man 页面。

- Libcap

这个软件包中，包含实现了可以用于 Linux 内核的，从用户空间到 POSIX 1003.1e 的接口。

- Libelf

elfutils 项目为 ELF 文件和 DWARF 数据提供了库和工具。该软件包中的大部分实用程序都可以在别的软件包中使用，但是该库需求使用默认（且最有效的）配置构建 Linux 内核。

- Libffi

这个软件包实现了一个可移植的、高级编程接口，以适应各种调用惯例。在编译时，一些程序可能还不知道给函数传递的是什么参数。例如，一个解释器可能会在运行时才被告知函数调用中所传递的参数的个数和类型。Libffi 可以应用于这些程序，充当从解释程序到被编译代码的桥梁。

- Libpipeline

这个软件包中，包含一个以灵活和便捷的方式操作子进程流水线的库。Man-DB 软件包会用到。

- Libtool

这个软件包中，包含了一些 GNU 通用库支持脚本。它降低了在一致、可移植的接口上使用共享库的复杂度。在其他 LFS 软件包的测试套件里需要它。

- Linux 内核

这个软件包就是操作系统。即我们常说的「GNU/Linux」中的「Linux」。

- M4

这个软件包中，包含一个普通的文本宏处理器，作为其它程序的构建工具使用。

- Make

这个软件包中，包含了一个指导软件包构建的程序。LFS 中的几乎每个包都需要它。

- Man-DB

这个软件包中，包含了一些查找和查看 man 页面的程序。由于其更好的国际化功能，用来代替 man 软件包。它提供了 man 程序。

- Man-pages

这个软件包中，包含基本的 Linux man 页面的真正内容。

- Meson

这个软件包提供了自动化构建软件的工具。Meson 的主要目标是最小化软件开发者，在其构建的系统上，花费的配置时间。

- MPC

这个软件包中，包含复数运算的函数。Gcc 需要它。

- MPFR

这个软件包中，包含多精度运算的函数。Gcc 需要它。

- Ninja

这个软件包中，包含了一个专注于速度的小型构建系统。其输入文件由更高层次的构建系统生成，并且可能快速地运行构建过程。

- Ncurses

这个软件包中，包含了一些处理字符界面的不依赖特定终端的库。通常用来为菜单系统提供光标控制。一些 LFS 的软件包会用到。

- Openssl

这个软件包中，提供了与加密相关管理工具和库。提供的这些加密功能对于他软件包（包括 Linux 内核）而言非常有用。

- Patch

这个软件包中，包含了一个通过 patch 文件来修改或新建文件的程序，patch 文件通常是由 diff 程序创建的。一些 LFS 软件包的构建过程会需要它。

- Perl

这个软件包中，包含了一个运行时语言 PERL 的解析器。一些 LFS 软件包的安装和测试套件会需要它。

- Pkg-config

这个软件包提供了一个返回已安装库或软件包的元数据的程序。

- Procps-NG

这个软件包中，包含了一些监视进程的程序。这些程序对系统管理非常有用，也被用于 LFS 的启动脚本。

- Psmisc

这个软件包中，包含了一些显示运行中进程信息的程序。这些程序对系统管理非常有用。

- Python 3

这个软件包提供了一种解释型语言，该语言的设计理念强调代码的可读性。

- Readline

这个软件包提供了一些命令行编辑和历史功能的库。Bash 会使用它。

- Sed

这个软件包提供了不通过文本编辑器而直接编辑文本的功能。大部分 LFS 软件包的配置脚本需要它。

- Shadow

这个软件包中，包含了一些以安全方式处理密码的程序。

- systemd

作为 Sysvinit 的替代品，这个包提供了 init 程序，以及一些其它的启动和系统控制的功能。很多 Linux 商业发行版都用它。

- Tar

这个软件包提供了归档和提取 LFS 中的几乎所有软件包的能力。

- Tcl

这个软件包中，包含了在很多 LFS 软件包测试套件中使用的工具命令语言。仅安装在临时工具链中。

- Texinfo

这个软件包中，包含了一些读、写以及转换信息页面的程序。在很多 LFS 软件包的安装过程中会使用它。

- Util-linux

该软件包中，包含了许多工具。其中有处理文件系统、控制台、分区和消息的工具。

- Vim

这个软件包中，包含一个编辑器。由于 vi 编辑器的经典以及大量的强大功能而选择它。对很多用户来说一个编辑器是一个非常个人的选择，如果需要的话也可以选择其它编辑器。

- XML::Parser

这个软件包是和 Expat 交互的 Perl 模块。

- XZ Utils

这个软件包中，包含了一些压缩和解压缩文件的程序。通常它的压缩率最高，在解压 XZ 或者 LZMA 格式的软件包时非常有用。

- Zlib

这个软件包中，包含了一些程序所使用的压缩和解压缩功能。

## 前提条件

构建一个 LFS 系统并不是一个简单的任务。它要求对 Unix 系统管理有一定水平的了解，以便可以解决问题并正确地执行所列出的命令。特别是，至少你应该拥有使用命令行 (shell)，复制或移动文件和目录、列出目录和文件内容、切换当前目录的能力。同时也希望你拥有使用 and 安装 Linux 软件的基本知识。

因为 LFS 书中假设你至少有这些基本的技能，所以众多的 LFS 提供的论坛看起来在这方面也给你提供不了太多的帮助。你会发现对于这些基本知识的问题并不会得到解答，或者会简单的建议你去看一下 LFS 主要的提前阅读列表。

构建 LFS 系统之前，我们建议阅读以下内容：

- 构建软件 HOWTO <http://www.tldp.org/HOWTO/Software-Building-HOWTO.html>

这是一个在 Linux 上编译和安装「通用」Unix 软件包的综合指南。尽管成文已有一段时间，但其对于编译和安装软件中所需的基本技能，还算一个不错的总结。

- 初学者源码安装指南 <http://moi.vonos.net/linux/beginners-installing-from-source/>

该指南很好的总结了从源码构建软件时所需要的基础技能和技巧。

## 排版约定

为了能更容易理解，这里有一些全书使用的排版约定。本段，会包含一些来自 Linux From Scratch 中排版格式的示例。

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

这种形式的文本，设计出来就是为了让读者照着输入的，除非周围文本中另有说明。部分的解释段落和会用它，以指出引用的那条命令。（译注：由于文字排版原因，其中的内容即便是注释，也不会翻译，下同。）

某些情况下，会在一行的行末添加反斜杠，将一个逻辑行的内容分写成两个或更多的物理行。

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

注意，反斜杠的后面必须紧跟一个回车符。其它的空白字符，例如空格和 Tab 将导致错误的结果。

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

这种形式的文本（等宽文本）用于显示屏幕输出，通常是所运行命令的输出结果。这种形式也用于文件名的显示，例如：/etc/ld.so.conf。

强调

这种形式的文本在本书中有多重目的。主要目的是强调重要的内容或项目。

<http://www.linuxfromscratch.org/>

这种格式用来表示 LFS 社区内部及外部网页的超链接。包括 HOWTO，下载地址和网站等。

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

这种格式在创建配置文件中会使用。第一个命令告诉系统新建一个文件 `$LFS/etc/group`，然后后面的行中不论在输入了什么都会输入到文件中，直至遇到文件终结符 (EOF)。因此，这整个部分会如看到的那样输入。

```
<#####>
```

这种格式用来封装需要替换为合适内容的文本，或者复制粘贴操作。

```
[#####]
```

这种格式用来封装可选项文本。

```
passwd(5)
```

这种格式用来表示特定的手册 (man) 页面。括号内的数字表示该手册的特定部分。例如 `passwd` 有两个手册页面。在 LFS 安装说明中，这两个手册页面会表示为 `/usr/share/man/man1/passwd.1` 以及 `/usr/share/man/man5/passwd.5`。当书中使用 `passwd(5)` 时它特指 `/usr/share/man/man5/passwd.5`。`man passwd` 会输出它找到匹配「passwd」的第一个手册页面，也就是 `/usr/share/man/man1/passwd.1`。在这个例子中，你需要执行 `man 5 passwd` 才能阅读指定的手册页面。应该注意的是，大部分的手册页面在不同部分不会有重复的页面名字。因此，`man <program name>` 通常就足够了。

## 本书结构

本书分为以下几个部分。

### 第一部分 介绍

第一部分解释了一些安装 LFS 时的重要注意事项。同时还提供了本书的基本信息。

### 第二部分 构建的准备工作

第二部分描述了构建的一些准备工作，包括分区，下载软件包，编译一些临时工具。

### 第三部分 构建 LFS 系统

第三部分引导用户开始 LFS 系统的构建——逐一的编译和安装所有的软件包，设置启动脚本，安装内核。生成的 Linux 系统是继续编译其它一系列软件的基础，通过那些软件来扩展系统，系统才能更好地满足我们的需求。在本书的最后，我们还给出了一个便于使用的引用列表，包括安装好的程序、库和一些重要文件。

## 勘误表

构建 LFS 系统的软件是在不断的更新和改进的，也许在这本 LFS 书发布以后就会出现一些安全警告或者是修复了一些 bug。请在着手构建之前访问 <http://www.linuxfromscratch.org/lfs/errata/systemd/>，检查此版本 LFS 软件包的版本或说明，是否需要修改以解决安全漏洞或其它 bug。你应该关注本书的一切变化，且将其应用到 LFS 系统的构建中去。

如果 LFS 的中文手册翻译的有问题，或是你对 LFS 中文手册有自己的建议和意见。可以通过中文手册的 Github 库 LFS-BOOK <https://github.com/LCJT/LFS-BOOK> 找到并告诉我们。

# Part I. 介绍

# Chapter 1. 介绍

## 1.1. 如何构建 LFS 系统

LFS 系统需要在一个已经安装好的 Linux 发行版（比如 Debian、OpenMandriva、Fedora 或 OpenSUSE 等）中构建。这个已有的 Linux 系统（即宿主）作为构建新系统的起始点，提供了必要的程序，包括编译器、链接器和 shell。请在安装发行版的过程中选择「Development（开发）」选项以便使用这些开发工具。

你可以选择在电脑上安装一个独立的发行版，亦可以临时使用商业发行版的 LiveCD。

本手册的第 2 章 描述了如何创建一个新的 Linux 本地分区和文件系统。这就是编译和安装全新的 LFS 系统的地方。第 3 章 介绍了构建 LFS 系统所需要下载的软件包和补丁，以及如何将它们保存到新的文件系统中。第 4 章 讨论了如何设置恰当的工作环境。还请务必仔细阅读第 4 章，它会告诉您在继续 Chapter#5 以及后面的工作之前，尤其需要注意的几件事情。

Chapter#5 解释了为构建编译套件（工具链）需要安装的多个软件包，在 Chapter#6 中将会使用这套工具构建真正的系统。其中有一些包需要解决循环依赖——比如你需要一个编译器来编译出一个编译器。

Chapter#5 还展示了如何构建第一阶段的工具链，包括 Binutils 和 GCC（第一阶段的主要意义就是重新安装这两个核心软件包）。下一步将来构建 Glibc，即 C 语言库。Glibc 将使用第一阶段构建的工具链编译。然后，第二阶段的工具链编译就完成了。这次，工具链将会被动态链接到新建立的 Glibc。Chapter#5 中剩余的其他包都将使用第二阶段的工具链编译。当这些过程完成之后，除了正在运行的内核外，LFS 的安装过程就不再依赖于宿主发行版了。

把新系统从宿主发行版中分离出来的工作看起来可能有点多余。Section#5.2, “工具链技术说明”给出了为什么要这么做的完整技术说明。

在 Chapter#6 中，将构建完整的 LFS 系统。我们将会使用 chroot（切换到 root 用户的）命令来进入一个虚拟环境并启动一个新的 shell，且整个虚拟环境就将作为 LFS 分区的根目录。这个过程，和重启电脑并指示内核将 LFS 分区挂载为根分区十分相像。只是系统并没有真正重启，而是通过 chroot 实现的，而这样做的原因是想要实现一个真正可以启动的系统还有很多工作没有完成。使用「chrooting」最大的好处在于，构建 LFS 的同时可以继续使用宿主系统。在等待包编译的过程中，你可以继续正常的使用电脑。

在完成基本的安装过程之后，您还需要阅读第 7 章 对系统做简单的配置，然后阅读第 8 章 配置内核和引导器（bootloader）。第 9 章 包含了一些 LFS 后续使用的建议。完成本手册的所有步骤之后，就可以重启电脑，进入新的 LFS 系统了。

大致流程就是如此。每一步的细节都会在后续的章节中逐一阐明。在你开始 LFS 的旅途之后，混沌的过程便会渐渐明晰。

## 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:

- 
- Bash 5.0
- Binutils-2.32
- Bison-3.3.2
- DejaGNU-1.6.2
- D-Bus-1.12.12
- Diffutils-3.7
- E2fsprogs-1.44.5
- File-5.36
- GDBM-1.18.1
- Glibc-2.29
- Grep-3.3
- Groff-1.22.4
- Gzip-1.10
- IPRoute2-4.20.0



- Kmod-26
- Libcap-2.26
- Libelf-0.176 (from elfutils)
- Libpipeline-1.5.1
- Linux-4.20.12
- Man-DB-2.8.5
- Meson-0.49.2
- MPFR-4.0.2
- Ninja-1.9.0
- Openssl-1.1.1a
- Perl-5.28.1
- Psmisc-23.2
- Python-3.7.2
- Readline-8.0
- Sed-4.7
- systemd-240
- Tar-1.31
- Tcl-8.6.9
- Tzdata-2018i
- Util-Linux-2.33.1

Added:

- 

Removed:

- 

## 1.3. Changelog

This is version 8.4-systemd of the Linux From Scratch book, dated 2019 年 3 月 1 日. 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:

- 2019-03-01
  - [bdubbs] - LFS-8.4 released.
- 2019-02-25
  - [bdubbs] - Update to linux-4.20.12. Fixes #4425.
  - [bdubbs] - Update to elfutils-0.176. Fixes #4426.
  - [bdubbs] - Update to file-5.36. Fixes #4429.
  - [renodr] - Add a security patch for systemd-240 to fix a PID1 crash over D-Bus. Fixes #4428.
- 2019-02-19
  - [bdubbs] - Add an optional modification to the build procedure for ninja to allow use the the environment variable NINJAJOBS.
- 2019-02-14
  - [bdubbs] - Update to linux-4.20.8. Fixes #4423.
  - [bdubbs] - Fix a problem when building Python in Chapter 5 where some hosts may use host dependent headers.
- 2019-02-11
  - [bdubbs] - Update to linux-4.20.7. Fixes #4421.

- [bdubbs] - Update to kmod-26. Fixes #4422.
- 2019-02-08
  - [renodr] - Update host system requirements.
- 2019-02-06
  - [bdubbs] - Simlify instructions for glibc in Chapter 5. Thanks to Romain Geissler for the report.
- 2019-02-05
  - [bdubbs] - Update to bison-3.3.2. Fixes #4419.
  - [bdubbs] - Update to meson-0.49.2. Fixes #4420.
- 2019-02-02
  - [bdubbs] - Fix psmisc URL. Fixes #4418.
  - [bdubbs] - Update to binutils-2.32. Fixes #4417.
- 2019-02-01
  - [bdubbs] - Update to bison-3.3.1. Fixes #4412.
  - [bdubbs] - Update to glibc-2.29. Fixes #4415.
  - [bdubbs] - Update to libpipeline-1.5.1. Fixes #4413.
  - [bdubbs] - Update to linux-4.20.6. Fixes #4409.
  - [bdubbs] - Update to meson-0.49.1. Fixes #4410.
  - [bdubbs] - Update to mpfr-4.0.2. Fixes #4416.
  - [bdubbs] - Update to ninja-1.9.0. Fixes #4414.
- 2019-01-27
  - [pierre] - Fix a bug introduced in tar-1.31, by adding a sed to the build instructions. Also remove an obsolete comment about a failing test.
- 2019-01-20
  - [renodr] - Regenerate the systemd man pages tarball with the Non-namespaced versions of the Docbook XSL Stylesheets.
- 2019-01-11
  - [renodr] - Add a security patch for systemd-240. This fixes CVE-2018-16865 and CVE-2018-16864 (memory corruption in journald leading to stack overflows / arbitrary code execution). Apply this as soon as you can. Fixes #4408.
- 2019-01-10
  - [bdubbs] - Update to linux-4.20.1. Fixes #4398.
  - [bdubbs] - Update to diffutils-3.7. Fixes #4401.
  - [bdubbs] - Update to tar-1.31. Fixes #4402.
  - [bdubbs] - Update to man-db-2.8.5. Fixes #4403.
  - [bdubbs] - Update to bash-5.0. Fixes #4404.
  - [bdubbs] - Update to readline-8.0. Fixes #4405.
  - [bdubbs] - Update to iproute2-4.20.0. Fixes #4406.
  - [bdubbs] - Update to util-linux-2.33.1. Fixes #4407.
- 2019-01-01
  - [bdubbs] - Update to gzip-1.10. Fixes #4400.
  - [bdubbs] - Update to tzdata-2018i. Fixes #4399.
- 2018-12-27
  - [renodr] - Update to linux-4.19.12. Fixes #4389.
  - [renodr] - Update to e2fsprogs-1.44.5. Fixes #4390.
  - [renodr] - Update to bison-3.2.4. Fixes #4391.
  - [renodr] - Update to sed-4.7. Fixes #4392.
  - [renodr] - Update to grep-3.3. Fixes #4393.

- [renodr] - Update to systemd-240. Contains a critical fix for systemd-tmpfiles (privilege escalation). Fixes #4394.
- [renodr] - Update to Python-3.7.2. Fixes #4395.
- [renodr] - Update to groff-1.22.4. Fixes #4396.
- 2018-12-12
  - [renodr] - Add a note to libffi about optimizing for the specific CPU in use at compile time. Similar to GMP, this causes Illegal Operation errors if the installation is moved to another system.
- 2018-12-11
  - [bdubbs] - Update to dbus-1.12.12. Fixes #4385.
  - [bdubbs] - Update to linux-4.19.8. Fixes #4387.
  - [bdubbs] - Update to meson-0.49.0. Fixes #4388.
- 2018-12-01
  - [bdubbs] - Move /etc/bash\_completions.d/grub to a better location. Fixes #4385.
  - [bdubbs] - Update to dejagnu-1.6.2. Fixes #4382.
  - [bdubbs] - Update to linux-4.19.6. Fixes #4383.
  - [bdubbs] - Update to perl-5.28.1. Fixes #4384.
- 2018-11-25
  - [bdubbs] - Update to bison-3.2.2. Fixes #4380.
- 2018-11-24
  - [dj] - Update to linux-4.19.4. Fixes #4381.
  - [dj] - Update to systemd-239-6b4878d.
  - [dj] - Add "wheel" group to systemd groups.
  - [dj] - Add touch to the list of moved coreutils programs, and clarify necessity of the moves to meet FHS compliance.
- 2018-11-21
  - [renodr] - Add "wheel" group to satisfy systemd requirements. Fixes #4376.
  - [renodr] - Add a sed to fix a bug in autoconf's test suite. Fixes #4372.
  - [renodr] - Update to tcl-8.6.9. Security update. Fixes #4375.
  - [renodr] - Update to openssl-1.1.1a. This is a security update. Fixes #4379.
  - [renodr] - Update to systemd-239-25d1ba1. This fixes three security problems in systemd. Fixes #4377.
  - [renodr] - Update to linux-4.19.3. Fixes #4373.
  - [renodr] - Update to elfutils-0.175. Fixes #4374.
- 2018-11-19
  - [bdubbs] - Update to libcap-2.26. Fixes #4378.
- 2018-11-09
  - [bdubbs] - Update to meson-0.48.2. Fixes #4371.
  - [bdubbs] - Update to bison-3.2.1. Fixes #4370.
- 2018-11-06
  - [bdubbs] - Update to bison-3.2. Fixes #4367.
  - [bdubbs] - Update to linux-4.19.1. Fixes #4368.
  - [bdubbs] - Update to tzdata-2018g. Fixes #4366.
  - [bdubbs] - Update to util-linux-v2.33. Fixes #4353.
- 2018-10-29
  - [dj] - Update to gdbm-1.18.1. Fixes #4364.
  - [dj] - Update to Python-3.7.1. Fixes #4361.
- 2018-10-27
  - [dj] - Update to iproute2-4.19.0. Fixes #4363.

- [dj] - Update to file-5.35. Fixes #4359.
- [dj] - Update to tzdata-2018f. Fixes #4358.
- [dj] - Update to meson-0.48.1. Fixes #4357.
- [dj] - Update to linux-4.19. Fixes #4356.
- 2018-10-10
  - [dj] - Removed incorrect link to /tool/lib64 in systemd instructions. Fixes #4355.
  - [dj] - Added systemd-239-meson-0.48.0\_fixes-1.patch to resolve build errors with meson.
  - [dj] - Update to meson-0.48.0. Fixes #4351.
  - [dj] - Update to linux-4.18.12. Fixes #4352.
- 2018-09-30
  - [dj] - Restore build of Util-Linux in chapter5 to avoid reciprocal dependency for systemd.
  - [dj] - Moved installation of Util-Linux and E2fsprogs after Procps to satisfy build order in the systemd book. This has no effect on the SysV book.
- 2018-09-20
  - [bdubbs] - Clean up of unneeded symbolic links. Reordered packages so version specific packages are built as late as possible in Chapter 6. Now building util-linux in Chapter 5 is unneeded and has been removed. Fixes #4345 and #4349.
  - [bdubbs] - Update to elfutils-0.174 (libelf). Fixes #4348.
  - [bdubbs] - Update to psmisc-23.2. Fixes #4347.
  - [bdubbs] - Update to openssl-1.1.1. Fixes #4346.
  - [bdubbs] - Update to linux-4.18.9. Fixes #4344.
- 2018-09-02
  - [bdubbs] - Update to bison-3.1. Fixes #4342.
  - [bdubbs] - Update to meson-0.47.2. Fixes #4341.
  - [bdubbs] - Update to gdbm-1.18. Fixes #4340.
  - [bdubbs] - Update to e2fsprogs-1.44.4. Fixes #4338.
- 2018-09-01
  - [bdubbs] - LFS-8.3 released.

## 1.4. 资源

### 1.4.1. FAQ

如果在构建 LFS 系统的过程中遇到任何的错误，或者是有任何的疑问，亦或者认为本手册存在任何的拼写错误，请先阅读常见问题列表（FAQ）：<http://www.linuxfromscratch.org/faq/>。

### 1.4.2. 邮件列表

[linuxfromscratch.org](http://www.linuxfromscratch.org) 的服务器上部署了一些和 LFS 项目开发工作相关的邮件列表。其中包括主要开发列表和支持列表，以及一些其他相关话题的讨论列表。如果 FAQ 不能解决您的问题，那么您可以在这里搜索邮件列表：<http://www.linuxfromscratch.org/search.html>。

关于这些列表的订阅、归档、和其他的信息，都可以访问 <http://www.linuxfromscratch.org/mail.html> 获取。

### 1.4.3. IRC 频道

有些 LFS 的社区成员也会在 IRC 上为别人提供帮助。在使用这种方法之前，请保证问题在 LFS 的 FAQ 或者是邮件列表中没有提及。您可以通过 [irc.freenode.net](http://irc.freenode.net) 查找 #LFS-support 加入讨论。

### 1.4.4. 镜像站点

LFS 项目在世界范围内有许多镜像站点，方便大家访问我们的网站以及下载所需文件。请访问 LFS 站点 <http://www.linuxfromscratch.org/mirrors.html> 查看最新的镜像站点列表。

## 1.4.5. 联系信息

请直接通过某个 LFS 邮件列表（上文已列出）提出问题和评论。

## 1.5. 帮助

如果在使用本书的过程中有疑问或碰到问题，可以先去看下 FAQ 页面<http://www.linuxfromscratch.org/faq/#generalfaq>。那里已经解决了很多经常遇到的问题。如果你的问题在那里找不到答案，可以先尝试找出问题的原因。下面页面里的提示可以提供一些帮你定位问题的帮助：<http://www.linuxfromscratch.org/hints/downloads/files/errors.txt>。

如果在 FAQ 里找不到你遇到的问题，还可以在这个邮件列表里搜索一下：<http://www.linuxfromscratch.org/search.html>。

我们还有一个很棒的 LFS 社区，大家都乐意通过邮件列表和 IRC 提供协助（参看本书的 Section#1.4, “资源”）。不过，我们每天收到的支持问题中有很多其实可以通过查看 FAQ 和搜索邮件列表轻松解决。所以为了让我们能最大可能地提供更好的协助，希望你碰到问题能自己先研究一下。这样可以让我们有精力去关注更罕见问题。如果你自己搜索不到解决方式，请在你的帮助请求里收集所有相关信息（下面提到的）。

### 1.5.1. 需要提供的信息

除了对遇到的问题的简短描述外，帮助请求里还需要包含的以下的关键信息：

- 所用手册的版本（当前版本为 8.4-systemd）
- 用来构建 LFS 的宿主机器的 Linux 发行版以及版本
- 本手册 Host System Requirements 中脚本的所有输出信息
- 出现问题的软件包或本手册的章节
- 精确的错误信息或表现形式
- 注明你是否已经脱离了本书的内容

#### Note

脱离本手册的内容并不意味着我们就一定不会帮你。毕竟，LFS 还是属于个人爱好。坦率地告知别人对已验证的流程作出的任何改动，有助于我们评估和找到你问题的可能原因。

### 1.5.2. 配置脚本问题

如果在运行 configure 脚本时遇到问题，查看一下 config.log 文件。这个文件中可能会包含 configure 脚本运行时没有输出到屏幕上的错误信息。想寻求帮助的话请包含相关行。

### 1.5.3. 编译问题

屏幕上的显示输出和各个文件的内容都有助于定位编译发生问题的原因。configure 和 make 命令的输出信息都有用。然而通常并不需要在寻求帮助时将这些信息一股脑的贴出来，但是一定要包含足够的相关信息。下面的例子就是在执行 make 命令出错后应该贴出来的输出信息：

```
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
```

在此例子中，很多人可能仅仅就贴出最后的一部分：

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

这样的信息并不能用于诊断问题，实际上它仅仅是告知别人出了问题，却没有指出到底是哪里出现问题。这也就是为什么要提供更加完整的信息。比如在上面的例子中，就告诉别人所执行的命令及相应的错误信息。

这个链接 <http://catb.org/~esr/faqs/smart-questions.html> 中的文章告知人们如何在互联网上寻求帮助，可以去看一下，且遵循其中的提问原则进行提问，可以提高获得帮助的概率（亦让别人对你伸出援手更加方便）。

## Part II. 准备构建

## Chapter 2. 准备宿主系统

### 2.1. 简介

本章将检查宿主机上用于构建 LFS 的工具，没有的话，安装它们。然后，会准备用于安装 LFS 的分区。包括建立分区、为分区设置文件系统，挂载分区。

### 2.2. 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.

Earlier versions of the listed software packages may work, but have not been tested.

- Bash-3.2 (/bin/sh should be a symbolic or hard link to bash)
- Binutils-2.25 (Versions greater than 2.32 are not recommended as they have not been tested)
- Bison-2.7 (/usr/bin/yacc should be a link to bison or small script that executes bison)
- Bzip2-1.0.4
- Coreutils-6.9
- Diffutils-2.8.1
- Findutils-4.2.31
- Gawk-4.0.1 (/usr/bin/awk should be a link to gawk)
- GCC-5.2 including the C++ compiler, g++ (Versions greater than 8.2.0 are not recommended as they have not been tested)
- Glibc-2.11 (Versions greater than 2.29 are not recommended as they have not been tested)
- Grep-2.5.1a
- Gzip-1.3.12
- Linux Kernel-3.2

The reason for the kernel version requirement is that we specify that version when building glibc in Chapter#6 at the recommendation of the developers. It is also required by udev.

If the host kernel is earlier than 3.2 you will need to replace the kernel with a more up to date version. There are two ways you can go about this. First, see if your Linux vendor provides a 3.2 or later kernel package. If so, you may wish to install it. If your vendor doesn't offer an acceptable kernel package, or you would prefer not to install it, you can compile a kernel yourself. Instructions for compiling the kernel and configuring the boot loader (assuming the host uses GRUB) are located in 第 8 章.

- M4-1.4.10
- Make-4.0
- Patch-2.5.4
- Perl-5.8.8
- Python-3.4
- Sed-4.1.5
- Tar-1.22
- Texinfo-4.7
- Xz-5.0.0

#### Important

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
# Simple script to list version numbers of critical development tools
export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4
MYSH=$(readlink -f /bin/sh)
echo "/bin/sh -> $MYSH"
echo $MYSH | grep -q bash || echo "ERROR: /bin/sh does not point to bash"
unset MYSH

echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1

if [ -h /usr/bin/yacc ]; then
    echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
elif [ -x /usr/bin/yacc ]; then
    echo yacc is `/usr/bin/yacc --version | head -n1`
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 [ -h /usr/bin/awk ]; then
    echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
elif [ -x /usr/bin/awk ]; then
    echo awk is `/usr/bin/awk --version | head -n1`
else
    echo "awk not found"
fi
```

```
gcc --version | head -n1
g++ --version | head -n1
ldd --version | head -n1 | cut -d" " -f2- # glibc version
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`
python3 --version
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1 # texinfo version
xz --version | head -n1
```

```
echo 'int main(){}' > dummy.c && g++ -o dummy dummy.c
if [ -x dummy ]
then echo "g++ compilation OK";
else echo "g++ compilation failed"; fi
rm -f dummy.c dummy
EOF

bash version-check.sh
```

## 2.3. 分阶段构建 LFS

LFS 的设计是建立在一个会话中的。这也就是说假定系统在处理的过程中不会关机。但是实际上这并不意味着 LFS 的构建需要一气呵成。但是如果 LFS 恢复时的状态和之前不同，则某些程序必须在此之后重新构建。

### 2.3.1. 第 1-4 章

当在主机系统各种完成此章节的内容，重启时，需要注意一下的内容：

- 程序在 2.4 节以后都是以 root 用户完成的。所以需要重新将 LFS 的环境变量设置为 ROOT。

### 2.3.2. 第 5 章

- /mnt/lfs 分区必须挂载。
- 第 5 章涉及的所有指令都应该由 lfs 用户执行。所以在执行第 5 章的任务之前请先执行 `su - lfs`。
- Section#5.3, “通用编译指南”中提及的程序是十分重要的。如果在安装任何软件包时存在问题，请确保所有之前已经解压的软件包已经删除。重新将它们解压后再完成此章节中的所有命令。

### 2.3.3. 第 6-8 章

- /mnt/lfs 分区必须挂载。
- 当进入 chroot 环境，LFS 环境变量必须设置为 root 用户。否则不得使用 LFS 变量。
- 必须挂载虚拟文件系统。这可以在进入 chroot 之前或者之后通过改变主机的虚拟终端，以 root 用户执行 Section#6.2.2, “挂载和激活 /dev”和 Section#6.2.3, “挂载虚拟文件系统”中的命令。

## 2.4. 创建新分区

与绝大多数其它操作系统相同，安装 LFS 通常需要专门的分区。构建 LFS 系统比较推荐的方法是使用可用的空分区，或者如果条件允许，最好是在未分区的空间里新建分区。

最小化的系统需要大约 6 GB 的分区，这足以存储所有的源码包及满足编译的需求。但如果要将 LFS 作为主要的 Linux 系统，可能需要安装其它附加的软件，这将需要额外的空间。考虑到了日后所需的空间，一个 20 GB 的分区是比较合理的。LFS 系统本身并不会占用这么多的空间，但大分区将能提供充裕的临时储存空间，并为完成 LFS 以后添加附加功能留有余地。编译软件包可能需要较大的磁盘空间，但这些空间可以在软件包安装后回收。

由于编译过程中所需的内存 (RAM) 可能不足，用一个小型的磁盘分区作为 swap 空间是个不错的想法。内核会在此分区中储存较少使用的数据，从而为活动进程提供更多的内存。LFS 系统可以与宿主系统共用 swap 分区，这样就没有必要再新建一个了。

启动磁盘分区程序，通过如 `cfdisk` 或 `fdisk` 加上新分区所在的磁盘名——例如 `/dev/sda` 若是主盘为 IDE 的话。你需要创建一个 Linux 本地分区，并按需创建 swap 分区。如果你还不知道如何使用这些程序，请参考 `cfdisk(8)` 或 `fdisk(8)`。

### Note

对于有经验的用户，也可自行定制分区分案。新版 LFS 系统支持软件 RAID 阵列或 LVM 然而，这些方案需要用到 `initramfs`，这涉及到比较复杂的话题。并不建议首次尝试 LFS 的用户使用这样的分区分案。

请记住新分区的位置 (例如，`sda5`)。本书中将称其为 LFS 分区。还需要记住 swap 分区的位置。这些将会在 `/etc/fstab` 文件中用到。

## 2.4.1. 其它分区问题

LFS 邮件列表中经常有人问到关于系统分区的建议。这一话题非常主观。大多数发行版默认情况下会使用整个磁盘，仅为交换分区保留一小部分空间，但由于种种原因，这并不适合 LFS。这样做会降低灵活性，使得多个发行版或几个 LFS 版本之间共享数据变得困难，也让备份更耗时，还会导致文件系统结构分配不合理而浪费磁盘空间。

### 2.4.1.1. 根分区

为 LFS 根分区（不要与 `/root` 目录混淆）分配 10 GB 的空间是比较好的折中方案。这为构建 LFS 和大多数 BLFS 提供了足够的空间，也小到可以轻易创建多个分区用于实验。

### 2.4.1.2. 交换分区

大多数发行版会自动创建交换分区。一般来说，交换分区的推荐大小为物理内存的两倍左右，然而鲜少需要这样做。若是磁盘空间有限，可以设置为 2 GB 的交换分区并查看磁盘的交换量。

发生内存交换其实并不好。通常，你只需要观察磁盘活动以及系统对命令的响应程度就能知道这个系统是否在进行交换。通常在使用非常不合理的命令时才会发生交换，如尝试编辑一个大小为 5 GB 的文件时。如果交换在你的系统上是常态，那最好的办法就是为你的系统添置更多的物理内存。

### 2.4.1.3. Grub Bios 分区

如果是在 GPT (GUID Partition Table, GUID 分区表) 下分 boot 磁盘，那么必然会产生一个约 1 MB 左右的小分区，如果之前没有的话。这个分区可能还没有被格式化，但是它必须存在，GRUB 会在安装引导器的时候用到。如果使用 fdisk，该分区通常会被标记为 'BIOS Boot'。而如果使用的是 gdisk，则分区代码应为 EF02。

#### Note

Grub Bios 分区必须位于 BIOS 用于引导系统的磁盘驱动器上。却不一定要和 LFS 的根分区位于同一个磁盘驱动器。一个系统里的磁盘可能会用到不同类型的分区表。该磁盘仅与 boot 磁盘的分区表类型有关。

### 2.4.1.4. 常用分区

在分配磁盘时，有些分区不是必须的，但却值得你考虑。以下列表并不全面，仅供参考。

- `/boot` – 强烈推荐。此分区用于存储内核和其它启动信息。为了减少大容量磁盘启动时的潜在问题，尽量将该分区设为磁盘驱动器上第一个物理分区。100 MB 的空间就十分充裕了。
- `/home` – 强烈推荐。home 目录可用于跨发行版或多个 LFS 版本之间共享用户自定义内容。应该将尽量多的磁盘都分配给 home 分区。
- `/usr` – 独立的 `/usr` 分区常见于瘦客户端服务器或无盘工作站。LFS 通常不需要。5 GB 大小足以应付大部分安装。
- `/opt` – 该目录常用于在 BLFS 中，安装多个像 Gnome 或 KDE 这样无需将文件嵌入 `/usr` 层次结构的大型软件包。如果使用的话，5 到 10 GB 的空间就足够了。
- `/tmp` – 独立的 `/tmp` 分区是比较少见的，但这在配置瘦客户端时会有用。如果使用的话，很少超过几 GB。
- `/usr/src` – 该分区非常用于存储 BLFS 源文件并在构建不同版本的 LFS 中共享。它也可用于构建 BLFS 软件包。30-50 GB 的分区可以提供足够的空间。

任何你需要在启动时自动挂载的单独分区都需要写入到 `/etc/fstab` 文件中。有关如何指定分区的细节将在 Section#8.2, “创建 `/etc/fstab` 文件”中讨论。

## 2.5. 在分区上创建文件系统

现在已经有有了一个空的分区，可以创建文件系统了。LFS 可以使用 Linux 内核能识别的任何文件系统，但最常用的类型是 `ext3` 和 `ext4`。当然，至于具体采用哪种文件系统，应该取决于所要存储的文件特点和分区的大小。例如：

`ext2`

适用于那些分区容量不是太大，更新也不频繁的情况，例如 `/boot` 分区。

ext3

是 ext2 的改进版本，其支持日志功能，能够帮助系统从非正常关机导致的异常中恢复。是常用作一般需求的文件系统。

ext4

是 ext 文件系统的最新版。提供了很多新的特性，包括纳秒级时间戳、创建和使用巨型文件(16TB)、以及速度的提升。

其它文件系统，包括 FAT32、NTFS、ReiserFS、JFS 和 XFS 等都在专门领域有其独到的作用。关于这些文件系统更多的信息可以参考[http://en.wikipedia.org/wiki/Comparison\\_of\\_file\\_systems](http://en.wikipedia.org/wiki/Comparison_of_file_systems)。

LFS 假设要将根分区 (/) 的文件系统的类型设置为 ext4。运行如下指令，为 LFS 分区创建 ext4 文件系统：

```
mkfs -v -t ext4 /dev/<xxx>
```

请替换 <xxx> 为实际的 / 分区设备名。

如果你已经有了现成的 swap 分区,不需要重新格式化。如果是新建的 swap 分区，需要用下面的命令初始化：

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

请替换 <yyy> 为实际的 swap 分区设备名。

## 2.6. 设置 \$LFS 变量

在整本书里，会多次用到环境变量 LFS。你应该确保这个变量在整个 LFS 构建过程中总是定义了。它应该被设置为你将要构建的 LFS 系统的目录名——这里我们使用 /mnt/lfs 作为例子，但是选择哪一个目录是你的自由。如果你把 LFS 构建到一个单独的分区里，这个目录将成为那个分区的挂载点。选择一个本地目录并用以下命令设置该变量：

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

设置这个变量的好处在于有些命令可以直接写成类似 mkdir -v \$LFS/tools 这样。在处理这条命令时，会自动替换「\$LFS」为「/mnt/lfs」（或任何这个变量所指的地方）。

### Caution

不论何时，当你离开又重新进入这个工作环境时都不要忘了检查 LFS 是否设置（比如当你使用 su 切换到 root 或是另一个用户时）。使用如下命令检查 LFS 变量是否正确设置：

```
echo $LFS
```

请确保输出的是你构建 LFS 的那个目录的路径，如果你是按照例子设置的，那就是 /mnt/lfs。如果你输出的路径不正确，请使用本页前一部分提供的命令把 \$LFS 重新设置到正确的目录。

### Note

确保 LFS 变量一直为设置的一个方法是编辑你 home 目录下的 .bash\_profile 文件和 /root/.bash\_profile，并输入上述 export 命令。另外，在 /etc/passwd 文件中，为所有需要的用户指定 LFS 变量的 shell 需要是 bash，以确保 /root/.bash\_profile 文件作为登录过程的一部分，并入其中。

另一个考虑是登入宿主系统的方法。如果是通过图形显示管理器登入，那么就不会像寻常那样在虚拟终端启动时，采用用户的 .bash\_profile 文件。在这种情况下，需要将 export 命令添加至 root 用户的 .bashrc 中。另外，有些发行版还有指令，在非交互式 bash 的调用中不让 .bashrc 指令运行。在非交互式使用的测试前请保证 export 命令已经添加。

## 2.7. 挂载新分区

至此，文件系统已经创建妥当，下一步就是访问这些分区了。为此，需要将这些建立的分区挂载到选定的挂载点。本书假定的挂载点为环境变量 LFS 指向的地址，如前文所述。

创建挂载点并用下面的命令挂载 LFS 文件系统：

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
```

请用 LFS 分区替代 <xxx>。

如果 LFS 使用了多个分区（如，/ 和 /usr），需要挂载：

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext4 /dev/<yyy> $LFS/usr
```

请替换 <xxx> 和 <yyy> 为合适的分区名。

请确保在挂载新分区时没有使用过于严格的权限参数（如 nosuid 或 nodev 选项）。运行不带任何参数的 mount 命令来查看在挂载 LFS 分区时设置了什么参数。如果设置了 nosuid 和/或 nodev 参数，就需要重新挂载了。

## Warning

上述说明假定你在 LFS 的过程中不会重启你的计算机。如果你关闭了你的系统，你将需要在每次重新开始构建过程时重新挂载 LFS 分区，或者修改你宿主系统的 /etc/fstab 文件，让每次重新启动后都自动挂载它。示例如下：

```
/dev/<xxx> /mnt/lfs ext4 defaults 1 1
```

如果你还需使用其它可选分区，那就确保将它们也一并添加。

如果你要使用 swap 分区，请使用 swapon 命令确保它是可用的：

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

请替换 <zzz> 为 swap 分区名。

至此你已经建立了自己的工作空间，是时候下载软件包了。

## Chapter 3. 软件包和补丁

### 3.1. 简介

本章列出了一张软件包的清单，你需要下载它们来构建一个基础的 Linux 系统。列出的软件版本号便是该软件经过确认可以正常工作的版本，也是成书时我们使用的版本。我们强烈建议不要使用更新的版本，因为某个版本的编译指令并不一定适用于更新版本。最新的软件包会包含许多问题，需要特别对应。这些对应方法会在本书的开发版本中解决并固定下来。

我们无法保证下载的地址是一直有效的。如果在本书发布后下载地址变了，大部分软件包可以用 Google (<http://www.google.com/>) 解决。如果连搜索也失败了，那不妨试一试 <http://www.linuxfromscratch.org/lfs/packages.html#packages> 中提到的其他下载地址。

下载好的软件包和补丁需要保存在某处，这个某处最好是一个在整个构建过程中都能便捷地访问到的地方。另外还需要一个工作目录，用于解压并构建源码。`$LFS/sources` 可以同时作为软件包、补丁的存放所和工作目录。通过这个目录，所有需要的元素都将存储在 LFS 分区中，并且在整个构建过程中都可以访问。

在开始下载任务之前，先用 `root` 用户执行下面的命令，创建这个目录：

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

设置目录的写权限和粘滞模式。「粘滞模式」是指，即便多个用户对某个目录有写权限，但仅有文件的所有者，能在粘滞目录中删除该文件。运行以下命令将开启目录的写权限和粘滞模式：

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

一个简单的一口气下载所有软件包和补丁的方法是使用 `wget-list` 作为 `wget` 的输入。例如：

```
wget --input-file=wget-list --continue --directory-prefix=$LFS/sources
```

另外，从 LFS-7.0 开始，多了一个单独的文件 `md5sums`，可以在正式开始前校验所有的文件是否都正确。将这个文件复制到 `$LFS/sources` 目录中并执行：

```
pushd $LFS/sources
md5sum -c md5sums
popd
```

### 3.2. All Packages

Download or otherwise obtain the following packages:

- Acl (2.2.53) - 513 KB:

Download: <http://download.savannah.gnu.org/releases/acl/acl-2.2.53.tar.gz>

MD5 sum: 007aabf1dbb550bcdde52a244cd1070

- Attr (2.4.48) - 457 KB:

Home page: <https://savannah.nongnu.org/projects/attr>

Download: <http://download.savannah.gnu.org/releases/attr/attr-2.4.48.tar.gz>

MD5 sum: bc1e5cb5c96d99b24886f1f527d3bb3d

- Autoconf (2.69) - 1,186 KB:

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

Download: <http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz>

MD5 sum: 50f97f4159805e374639a73e2636f22e

- Automake (1.16.1) - 1,499 KB:

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

Download: <http://ftp.gnu.org/gnu/automake/automake-1.16.1.tar.xz>

MD5 sum: 53f38e7591fa57c3d2cee682be668e5b

- Bash (5.0) - 9,898 KB:

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

Download: <http://ftp.gnu.org/gnu/bash/bash-5.0.tar.gz>

MD5 sum: 2b44b47b905be16f45709648f671820b

- Bc (1.07.1) - 411 KB:

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

Download: <http://ftp.gnu.org/gnu/bc/bc-1.07.1.tar.gz>

MD5 sum: cda93857418655ea43590736fc3ca9fc

- Binutils (2.32) - 20,288 KB:

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

Download: <http://ftp.gnu.org/gnu/binutils/binutils-2.32.tar.xz>

MD5 sum: 0d174cdaf85721c5723bf52355be41e6

- Bison (3.3.2) - 2,060 KB:

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

Download: <http://ftp.gnu.org/gnu/bison/bison-3.3.2.tar.xz>

MD5 sum: c9b552dee234b2f6b66e56b27e5234c9

- Bzip2 (1.0.6) - 764 KB:

Download: <http://anduin.linuxfromscratch.org/LFS/bzip2-1.0.6.tar.gz>

MD5 sum: 00b516f4704d4a7cb50a1d97e6e8e15b

- Check (0.12.0) - 747 KB:

Home page: <https://libcheck.github.io/check>

Download: <https://github.com/libcheck/check/releases/download/0.12.0/check-0.12.0.tar.gz>

MD5 sum: 31b17c6075820a434119592941186f70

- Coreutils (8.30) - 5,234 KB:

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

Download: <http://ftp.gnu.org/gnu/coreutils/coreutils-8.30.tar.xz>

MD5 sum: ab06d68949758971fe744db66b572816

- D-Bus (1.12.12) - 2,029 KB:

Home page: <https://www.freedesktop.org/wiki/Software/dbus>

Download: <https://dbus.freedesktop.org/releases/dbus/dbus-1.12.12.tar.gz>

MD5 sum: ea11069521beeee4e47f0086596a43c8

- DejaGNU (1.6.2) - 514 KB:

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

Download: <http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.2.tar.gz>

MD5 sum: e1b07516533f351b3aba3423fafeffd6

- Diffutils (3.7) - 1,415 KB:

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

Download: <http://ftp.gnu.org/gnu/diffutils/diffutils-3.7.tar.xz>

MD5 sum: 4824adc0e95dbbf11dfbdfaad6a1e461

- E2fsprogs (1.44.5) - 7,448 KB:

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

Download: <https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.44.5/e2fsprogs-1.44.5.tar.gz>

MD5 sum: 8d78b11d04d26c0b2dd149529441fa80

- Elfutils (0.176) - 8,444 KB:

Home page: <https://sourceware.org/ftp/elfutils/>

Download: <https://sourceware.org/ftp/elfutils/0.176/elfutils-0.176.tar.bz2>

MD5 sum: 077e4f49320cad82bf17a997068b1db9

- Expat (2.2.6) - 502 KB:

Home page: <https://libexpat.github.io/>

Download: <https://prdownloads.sourceforge.net/expat/expat-2.2.6.tar.bz2>

MD5 sum: ca047ae951b40020ac831c28859161b2

- Expect (5.45.4) - 618 KB:

Home page: <https://core.tcl.tk/expect/>

Download: <https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz>

MD5 sum: 00fce8de158422f5ccd2666512329bd2



- File (5.36) - 856 KB:  
Home page: <https://www.darwinsys.com/file/>  
Download: <ftp://ftp.astron.com/pub/file/file-5.36.tar.gz>  
MD5 sum: 9af0eb3f5db4ae00fffc37f7b861575c

## Note

File (5.36) 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 (4.6.0) - 3,692 KB:  
Home page: <http://www.gnu.org/software/findutils/>  
Download: <http://ftp.gnu.org/gnu/findutils/findutils-4.6.0.tar.gz>  
MD5 sum: 9936aa8009438ce185bea2694a997fc1
- Flex (2.6.4) - 1,386 KB:  
Home page: <https://github.com/westes/flex>  
Download: <https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz>  
MD5 sum: 2882e3179748cc9f9c23ec593d6adc8d
- Gawk (4.2.1) - 2,916 KB:  
Home page: <http://www.gnu.org/software/gawk/>  
Download: <http://ftp.gnu.org/gnu/gawk/gawk-4.2.1.tar.xz>  
MD5 sum: 95cf553f50ec9f386b5dfcd67f30180a
- GCC (8.2.0) - 61,974 KB:  
Home page: <https://gcc.gnu.org/>  
Download: <http://ftp.gnu.org/gnu/gcc/gcc-8.2.0/gcc-8.2.0.tar.xz>  
MD5 sum: 4ab282f414676496483b3e1793d07862
- GDBM (1.18.1) - 920 KB:  
Home page: <http://www.gnu.org/software/gdbm/>  
Download: <http://ftp.gnu.org/gnu/gdbm/gdbm-1.18.1.tar.gz>  
MD5 sum: 988dc82182121c7570e0cb8b4fcd5415
- Gettext (0.19.8.1) - 7,041 KB:  
Home page: <http://www.gnu.org/software/gettext/>  
Download: <http://ftp.gnu.org/gnu/gettext/gettext-0.19.8.1.tar.xz>  
MD5 sum: df3f5690eaa30fd228537b00cb7b7590
- Glibc (2.29) - 16,129 KB:  
Home page: <http://www.gnu.org/software/libc/>  
Download: <http://ftp.gnu.org/gnu/glibc/glibc-2.29.tar.xz>  
MD5 sum: e6c279d5b2f0736f740216f152acf974
- GMP (6.1.2) - 1,901 KB:  
Home page: <http://www.gnu.org/software/gmp/>  
Download: <http://ftp.gnu.org/gnu/gmp/gmp-6.1.2.tar.xz>  
MD5 sum: f58fa8001d60c4c77595fbbb62b63c1d
- Gperf (3.1) - 1,188 KB:  
Home page: <http://www.gnu.org/software/gperf/>  
Download: <http://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz>  
MD5 sum: 9e251c0a618ad0824b51117d5d9db87e
- Grep (3.3) - 1,440 KB:  
Home page: <http://www.gnu.org/software/grep/>  
Download: <http://ftp.gnu.org/gnu/grep/grep-3.3.tar.xz>  
MD5 sum: 05d0718a1b7cc706a4bdf8115363f1ed

- Groff (1.22.4) - 4,044 KB:  
Home page: <http://www.gnu.org/software/groff/>  
Download: <http://ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz>  
MD5 sum: 08fb04335e2f5e73f23ea4c3adbf0c5f
- GRUB (2.02) - 5,970 KB:  
Home page: <http://www.gnu.org/software/grub/>  
Download: <https://ftp.gnu.org/gnu/grub/grub-2.02.tar.xz>  
MD5 sum: 8a4a2a95aac551fb0fba860ceabfa1d3
- Gzip (1.10) - 757 KB:  
Home page: <http://www.gnu.org/software/gzip/>  
Download: <http://ftp.gnu.org/gnu/gzip/gzip-1.10.tar.xz>  
MD5 sum: 691b1221694c3394f1c537df4eee39d3
- Iana-Etc (2.30) - 201 KB:  
Home page: <http://freecode.com/projects/iana-etc>  
Download: <http://anduin.linuxfromscratch.org/LFS/iana-etc-2.30.tar.bz2>  
MD5 sum: 3ba3afb1d1b261383d247f46cb135ee8
- Inetutils (1.9.4) - 1,333 KB:  
Home page: <http://www.gnu.org/software/inetutils/>  
Download: <http://ftp.gnu.org/gnu/inetutils/inetutils-1.9.4.tar.xz>  
MD5 sum: 87fef1fa3f603aef11c41dcc097af75e
- Intlttool (0.51.0) - 159 KB:  
Home page: <https://freedesktop.org/wiki/Software/intlttool>  
Download: <https://launchpad.net/intlttool/trunk/0.51.0/+download/intlttool-0.51.0.tar.gz>  
MD5 sum: 12e517cac2b57a0121cda351570f1e63
- IPRoute2 (4.20.0) - 691 KB:  
Home page: <https://www.kernel.org/pub/linux/utils/net/iproute2/>  
Download: <https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-4.20.0.tar.xz>  
MD5 sum: f3dab4c812812bbb5873cb90f471bcbf
- Kbd (2.0.4) - 1,008 KB:  
Home page: <http://ftp.altlinux.org/pub/people/legion/kbd>  
Download: <https://www.kernel.org/pub/linux/utils/kbd/kbd-2.0.4.tar.xz>  
MD5 sum: c1635a5a83b63aca7f97a3eab39ebaa6
- Kmod (26) - 540 KB:  
Download: <https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-26.tar.xz>  
MD5 sum: 1129c243199bdd7db01b55a61aa19601
- Less (530) - 332 KB:  
Home page: <http://www.greenwoodsoftware.com/less/>  
Download: <http://www.greenwoodsoftware.com/less/less-530.tar.gz>  
MD5 sum: 6a39bccf420c946b0fd7ffc64961315b
- Libcap (2.26) - 66 KB:  
Home page: <https://sites.google.com/site/fullycapable/>  
Download: <https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.26.tar.xz>  
MD5 sum: 968ac4d42a1a71754313527be2ab5df3
- Libffi (3.2.1) - 920 KB:  
Home page: <https://sourceware.org/libffi/>  
Download: <ftp://sourceware.org/pub/libffi/libffi-3.2.1.tar.gz>  
MD5 sum: 83b89587607e3eb65c70d361f13bab43
- Libpipeline (1.5.1) - 965 KB:  
Home page: <http://libpipeline.nongnu.org/>  
Download: <http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.1.tar.gz>  
MD5 sum: 4c8fe6cd85422baafd6e060f896c61bc
- Libtool (2.4.6) - 951 KB:  
Home page: <http://www.gnu.org/software/libtool/>  
Download: <http://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz>  
MD5 sum: 1bfb9b923f2c1339b4d2ce1807064aa5

- Linux (4.20.12) - 101,841 KB:  
Home page: <https://www.kernel.org/>  
Download: <https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.20.12.tar.xz>  
MD5 sum: edd3015435d60598b99cf6aaf223710e

## Note

The Linux kernel is updated relatively often, many times due to discoveries of security vulnerabilities. The latest available 4.20.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 (1.4.18) - 1,180 KB:  
Home page: <http://www.gnu.org/software/m4/>  
Download: <http://ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz>  
MD5 sum: 730bb15d96fffe47e148d1e09235af82
- Make (4.2.1) - 1,375 KB:  
Home page: <http://www.gnu.org/software/make/>  
Download: <http://ftp.gnu.org/gnu/make/make-4.2.1.tar.bz2>  
MD5 sum: 15b012617e7c44c0ed482721629577ac
- Man-DB (2.8.5) - 1,746 KB:  
Home page: <https://www.nongnu.org/man-db/>  
Download: <http://download.savannah.gnu.org/releases/man-db/man-db-2.8.5.tar.xz>  
MD5 sum: c5c6c3434be14a5527d43b5ad0f09a13
- Man-pages (4.16) - 1,592 KB:  
Home page: <https://www.kernel.org/doc/man-pages/>  
Download: <https://www.kernel.org/pub/linux/docs/man-pages/man-pages-4.16.tar.xz>  
MD5 sum: ad9f1ff81276fe8d90d077484d6d4b5e
- Meson (0.49.2) - 1,310 KB:  
Home page: <https://mesonbuild.com>  
Download: <https://github.com/mesonbuild/meson/releases/download/0.49.2/meson-0.49.2.tar.gz>  
MD5 sum: 0267b0871266056184c484792572c682
- MPC (1.1.0) - 685 KB:  
Home page: <http://www.multiprecision.org/>  
Download: <https://ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz>  
MD5 sum: 4125404e41e482ec68282a2e687f6c73
- MPFR (4.0.2) - 1,409 KB:  
Home page: <https://www.mpfr.org/>  
Download: <http://www.mpfr.org/mpfr-4.0.2/mpfr-4.0.2.tar.xz>  
MD5 sum: 320fbc4463d4c8cb1e566929d8adc4f8
- Ninja (1.9.0) - 187 KB:  
Home page: <https://ninja-build.org/>  
Download: <https://github.com/ninja-build/ninja/archive/v1.9.0/ninja-1.9.0.tar.gz>  
MD5 sum: f340be768a76724b83e6daab69009902
- Ncurses (6.1) - 3,288 KB:  
Home page: <http://www.gnu.org/software/ncurses/>  
Download: <http://ftp.gnu.org/gnu/ncurses/ncurses-6.1.tar.gz>  
MD5 sum: 98c889aaf8d23910d2b92d65be2e737a
- OpenSSL (1.1.1a) - 8,160 KB:  
Home page: <https://www.openssl.org/>  
Download: <https://openssl.org/source/openssl-1.1.1a.tar.gz>  
MD5 sum: 963deb2272d6be7d4c2458afd2517b73

- Patch (2.7.6) - 766 KB:  
 Home page: <https://savannah.gnu.org/projects/patch/>  
 Download: <http://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz>  
 MD5 sum: 78ad9937e4caadcba1526ef1853730d5
- Perl (5.28.1) - 12,083 KB:  
 Home page: <https://www.perl.org/>  
 Download: <https://www.cpan.org/src/5.0/perl-5.28.1.tar.xz>  
 MD5 sum: fbb590c305f2f88578f448581b8cf9c4
- Pkg-config (0.29.2) - 1,970 KB:  
 Home page: <https://www.freedesktop.org/wiki/Software/pkg-config>  
 Download: <https://pkg-config.freedesktop.org/releases/pkg-config-0.29.2.tar.gz>  
 MD5 sum: f6e931e319531b736fadc017f470e68a
- Procps (3.3.15) - 884 KB:  
 Home page: <https://sourceforge.net/projects/procps-ng>  
 Download: <https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.15.tar.xz>  
 MD5 sum: 2b0717a7cb474b3d6dfdeedfbad2ecc
- Psmisc (23.2) - 297 KB:  
 Home page: <http://psmisc.sourceforge.net/>  
 Download: <https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.2.tar.xz>  
 MD5 sum: 0524258861f00be1a02d27d39d8e5e62
- Python (3.7.2) - 16,648 KB:  
 Home page: <https://www.python.org/>  
 Download: <https://www.python.org/ftp/python/3.7.2/Python-3.7.2.tar.xz>  
 MD5 sum: df6ec36011808205beda239c72f947cb
- Python Documentation (3.7.2) - 6,072 KB:  
 Download: <https://docs.python.org/ftp/python/doc/3.7.2/python-3.7.2-docs-html.tar.bz2>  
 MD5 sum: 107ade7bb17efd104a22b2d457f4cb67
- Readline (8.0) - 2,907 KB:  
 Home page: <https://tiswww.case.edu/php/chet/readline/rltop.html>  
 Download: <http://ftp.gnu.org/gnu/readline/readline-8.0.tar.gz>  
 MD5 sum: 7e6c1f16aee3244a69aba6e438295ca3
- Sed (4.7) - 1,268 KB:  
 Home page: <http://www.gnu.org/software/sed/>  
 Download: <http://ftp.gnu.org/gnu/sed/sed-4.7.tar.xz>  
 MD5 sum: 777ddfd9d71dd06711fe91f0925e1573
- Shadow (4.6) - 1,639 KB:  
 Download: <https://github.com/shadow-maint/shadow/releases/download/4.6/shadow-4.6.tar.xz>  
 MD5 sum: b491fecbf1232632c32ff8f1437fd60e
- systemd (240) - 7,412 KB:  
 Home page: <https://www.freedesktop.org/wiki/Software/systemd/>  
 Download: <https://github.com/systemd/systemd/archive/v240/systemd-240.tar.gz>  
 MD5 sum: 0e4f91b513d4b04e2c10a5173e5a87b2
- systemd Man Pages(240) - 460 KB:  
 Home page: <https://www.freedesktop.org/wiki/Software/systemd/>  
 Download: <http://anduin.linuxfromscratch.org/LFS/systemd-man-pages-240.tar.xz>  
 MD5 sum: ca49a25e1cf330b02adc07218f430dae

## Note

The Linux From Scratch team generates its own tarball of the man pages using the systemd source. This is done in order to avoid unnecessary dependencies.

- Tar (1.31) - 2,052 KB:  
Home page: <http://www.gnu.org/software/tar/>  
Download: <http://ftp.gnu.org/gnu/tar/tar-1.31.tar.xz>  
MD5 sum: bc9a89da1185ceb2210de12552c43ce2
  - Tcl (8.6.9) - 9,772 KB:  
Home page: <http://tcl.sourceforge.net/>  
Download: <https://downloads.sourceforge.net/tcl/tcl8.6.9-src.tar.gz>  
MD5 sum: aa0a121d95a0e7b73a036f26028538d4
  - Texinfo (6.5) - 4,399 KB:  
Home page: <http://www.gnu.org/software/texinfo/>  
Download: <http://ftp.gnu.org/gnu/texinfo/texinfo-6.5.tar.xz>  
MD5 sum: 3715197e62e0e07f85860b3d7aab55ed
  - Time Zone Data (2018i) - 369 KB:  
Home page: <https://www.iana.org/time-zones>  
Download: <https://www.iana.org/time-zones/repository/releases/tzdata2018i.tar.gz>  
MD5 sum: b3f0a1a789480a036e58466cd0702477
  - Util-linux (2.33.1) - 4,542 KB:  
Home page: <http://freecode.com/projects/util-linux>  
Download: <https://www.kernel.org/pub/linux/utils/util-linux/v2.33/util-linux-2.33.1.tar.xz>  
MD5 sum: 6fcfea2043b5ac188fd3eed56aeb5d90
  - Vim (8.1) - 10,995 KB:  
Home page: <https://www.vim.org>  
Download: <ftp://ftp.vim.org/pub/vim/unix/vim-8.1.tar.bz2>  
MD5 sum: 1739a1df312305155285f0cfa6118294
  - XML::Parser (2.44) - 232 KB:  
Home page: <https://github.com/chorny/XML-Parser>  
Download: <https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.44.tar.gz>  
MD5 sum: af4813fe3952362451201ced6fbce379
  - Xz Utils (5.2.4) - 1030 KB:  
Home page: <https://tukaani.org/xz>  
Download: <https://tukaani.org/xz/xz-5.2.4.tar.xz>  
MD5 sum: 003e4d0b1b1899fc6e3000b24feddf7c
  - Zlib (1.2.11) - 457 KB:  
Home page: <https://www.zlib.net/>  
Download: <https://zlib.net/zlib-1.2.11.tar.xz>  
MD5 sum: 85adef240c5f370b308da8c938951a68
- Total size of these packages: about 381 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:

- Bzip2 Documentation Patch - 1.6 KB:  
Download: [http://www.linuxfromscratch.org/patches/lfs/8.4/bzip2-1.0.6-install\\_docs-1.patch](http://www.linuxfromscratch.org/patches/lfs/8.4/bzip2-1.0.6-install_docs-1.patch)  
MD5 sum: 6a5ac7e89b791aae556de0f745916f7f
- Coreutils Internationalization Fixes Patch - 168 KB:  
Download: <http://www.linuxfromscratch.org/patches/lfs/8.4/coreutils-8.30-i18n-1.patch>  
MD5 sum: a9404fb575dfd5514f3c8f4120f9ca7d
- Glibc FHS Patch - 2.8 KB:  
Download: <http://www.linuxfromscratch.org/patches/lfs/8.4/glibc-2.29-fhs-1.patch>  
MD5 sum: 9a5997c3452909b1769918c759eff8a2
- Kbd Backspace/Delete Fix Patch - 12 KB:  
Download: <http://www.linuxfromscratch.org/patches/lfs/8.4/kbd-2.0.4-backspace-1.patch>  
MD5 sum: f75cca16a38da6caa7d52151f7136895

- systemd security patch - 14 KB:

Download: [http://www.linuxfromscratch.org/patches/lfs/8.4/systemd-240-security\\_fixes-2.patch](http://www.linuxfromscratch.org/patches/lfs/8.4/systemd-240-security_fixes-2.patch)

MD5 sum: 10abebce8ff5d9fd402623ace39b5ab8

Total size of these patches: about 198.4 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/downloads/> and acquire any additional patches to suit your system needs.

## Chapter 4. 最后的准备工作

### 4.1. 简介

在本章，我们还需要为构建临时系统做一些额外的准备工作。我们会在 `$LFS` 中新建一个用于安装临时工具的目录，增加一个非特权用户用于降低风险，并为该用户创建合适的构建环境。我们还会解释用于测量构建 LFS 软件包花费时间的单位，或称「SBU」，并给出一些关于软件包测试套件的信息。

### 4.2. 创建目录 `$LFS/tools`

所有 Chapter#5 中编译的软件都会安装到 `$LFS/tools` 中，以确保和 Chapter#6 中编译的软件相互分离。这里编译的软件是临时工具，并不会成为最终的 LFS 系统中的一部分。将这些软件保存在单独的目录中，用完后方可弃置。这样做也可以防止这些程序在宿主机生成目录中突然停止工作（在 Chapter#5 中很容易发生意外）。

以 `root` 用户运行以下的命令来创建需要的文件夹：

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

下一步是在宿主系统中创建 `/tools` 的符号链接。将其指向 LFS 分区中新建的目录。同样以 `root` 用户运行下面的命令：

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

#### Note

上面的命令是正确的。ln 命令有一些语法变种，所以在报出你觉得可能是错误的信息之前检查一下 `info coreutils ln` 和 `ln(1)`。

创建的符号链接使得编译的工具链总是指向 `/tools` 文件夹，也就是说编译器、汇编器以及链接器无论是在第五章中（我们仍然使用宿主机的一些工具的时）还是之后（当我们「chrooted」到 LFS 分区时）都可以工作。

### 4.3. 添加 LFS 用户

当以 `root` 用户登录时，犯一个小错误可能会破坏或摧毁整个系统。因此，我们建议在本章中以非特权用户编译软件包。你当然可以使用你自己的用户名，但为了使其更容易建立一个干净的工作环境，创建一个名为 `lfs` 的新用户作为新组（同样命名为 `lfs`）的成员，并在安装过程中使用这个用户。以 `root` 用户运行以下命令来添加新用户：

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

命令行选项释义：

`-s /bin/bash`

将 `bash` 设置为 `lfs` 用户的默认 shell。

`-g lfs`

这个选项将用户 `lfs` 添加到组 `lfs` 中。

`-m`

为 `lfs` 用户创建主目录。

`-k /dev/null`

这个参数通过改变输入位置为特殊的空（null）设备，以防止可能从框架目录（默认是 `/etc/skel`）复制文件。

`lfs`

这是创建的组和用户的实际名称。

要以 `lfs` 用户身份登录（相较于以 `root` 身份登录的情况下切换到 `lfs` 用户时，无需为 `lfs` 用户设置密码），需要给 `lfs` 用户一个密码：

```
passwd lfs
```

通过更改目录所有者为 `lfs`，为用户 `lfs` 赋予了访问 `$LFS/tools` 目录的所有权限：

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

如果你按照建议创建了单独的工作目录，给 `lfs` 用户赋予这个目录的所有权：

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

下一步，以 `lfs` 用户身份登录。可能通过一个虚拟控制台、显示控制器，或者下面的切换用户命令完成：

```
su - lfs
```

这个「-」授意 `su` 启动登录 shell，而非 non-login shell。关于这两种 shell 类型的区别，可以在 `bash(1)` 和 `info bash` 中查看详细详情。

## 4.4. 设置环境

通过为 `bash` shell 创建两个开机启动的文件，设置合适的工作环境。当以 `lfs` 用户身份登录时，运行以下命令创建一个新的 `.bash_profile` 文件：

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

当以 `lfs` 用户身份登录时，初始 shell 通常是一个 login 的 shell，它先读取宿主机的 `/etc/profile` 文件（很可能包括一些设定和环境变量），然后是 `.bash_profile` 文件。`.bash_profile` 中的命令 `exec env -i.../bin/bash` 用一个除了 `HOME`，`TERM` 和 `PS1` 变量外，其他环境完全为空的新 shell 代替运行中的 shell。这能确保不会有潜在的和意想不到的危险环境变量，从宿主机泄露到构建环境中。这样做主要是为了确保环境的干净。

新的 shell 实例是一个 non-login 的 shell，不会读取 `/etc/profile` 或者 `.bash_profile` 文件，而是读取 `.bashrc`。现在，创建 `.bashrc` 文件：

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
EOF
```

`set +h` 命令关闭了 `bash` 的哈希功能。哈希通常是个好用的功能——`bash` 用一个哈希表来记录可执行文件的完整路径，以规避对 `PATH` 进行检索的时间和对同一个可执行文件的重复寻找。然而，新工具在安装后，应马上使用。通过关闭哈希功能，程序执行的时候就会一直搜索 `PATH`。如此，新编译的工具一旦可用，shell 便能在马上在文件夹 `$LFS/tools` 中找到它们，而不会去记录存在于不同地方的旧版该程序。

设置用户文件新建时的掩码（`umask`）为 `022`，以确保新建的文件和目录只有其所有者可写，但任何人都可读可执行（假设系统调用的 `open(2)` 使用的是默认模式，新文件将使用 `664` 权限模式、文件夹为 `755` 模式）。

`LFS` 变量应设置成选定的挂载点。

`LC_ALL` 变量控制某些程序的本地化，使它们的消息遵循特定国家的惯例。设置 `LC_ALL` 为「`POSIX`」或「`C`」（两者是等价的），确保在 `chroot` 环境中一切能如期望的那样进行。

`LFS_TGT` 变量设置了一个虽非默认，但在构建交叉编译器、连接器和交叉编译临时工作链时，用得上到的兼容的机器说明。Section#5.2，“工具链技术说明中包含更多信息。

通过把 `/tools/bin` 放在标准 `PATH` 变量的前面，使得所有在 Chapter#5 中安装的程序，一经安装 shell 便能马上使用。与之配合的关闭哈希功能，能在第五章环境中的程序在可用的情况下，限制使用宿主机中旧程序的风险。

最后，启用刚才创建的用户配置，为构建临时工具完全准备好环境：

```
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. 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.

### Note

For many modern systems with multiple processors (or cores) the compilation time for a package can be reduced by performing a "parallel make" by either setting an environment variable or telling the make program how many processors are available. For instance, a Core2Duo can support two simultaneous processes with:

```
export MAKEFLAGS='-j 2'
```

or just building with:

```
make -j2
```

When multiple processors are used in this way, the SBU units in the book will vary even more than they normally would. In some cases, the make step will simply fail. Analyzing the output of the build process will also be more difficult because the lines of different processes will be interleaved. If you run into a problem with a build step, revert back to a single processor build to properly analyze the error messages.

## 4.6. 关于测试套件

很多软件包都提供相应的测试套件。为新构建的软件包运行测试套件是非常好的习惯，因为这样做可以「保证」所有功能都已编译正确。经由一系列的测试，套件往往能够检查出软件包的功能是否都如开发人员预想的那样。然而，这并不能保证所测试的软件包万无一失。

有一些测试套件要相较而言更为重要。例如，核心工具链软件包——GCC, Binutils 和 Glibc——对于对于一个系统的正常运转起到至关重要的作用。要完成 GCC 和 Glibc 的测试套件可能要花费很长的时间，特别是对于硬件比较慢的设备来说，但还是强烈推荐完成它们！

### Note

经验表明，在 Chapter#5中运行测试套件并不是什么好主意。在那章有何无法回避的现实，就是宿主主机或多或少会对测试产生影响，经常导致一些令人摸不着头脑的错误信息。因为在 Chapter#5中构建的这些工具只是零时的，最终我们并不需要它们，所以我们并不推荐普通读者在 Chapter#5中运行测试套件。虽然为测试者和开发者提供了测试套件的说明，但是这依旧是可选项。

对 Binutils 和 GCC 执行测试套件时可能会使伪终端 (PTYs) 耗尽。造成大量的测试失败。造成问题的原因有很多，但最有可能的原因是宿主系统没能正确设置 `devpts` 文件系统。针对这个问题在 <http://www.linuxfromscratch.org/lfs/faq.html#no-ptys> 中有更详尽的讨论。

还有一些测试套件运行错误，是开发人员已知且视为不重要的。查看 <http://www.linuxfromscratch.org/lfs/build-logs/8.4/> 中的日志，确认这些失败信息是否都是意料之中的。该网址中涉及的内容会贯穿全书所有的测试。

# Chapter 5. Constructing a Temporary System

## 5.1. 简介

本章将向您展示如何构造一个最小的 Linux 系统。该系统将包含刚好足够构建 Chapter#6 中最终 LFS 系统所需的工具，以及一个比最小环境具有更好用户便利性的工作环境。

构建这个最小系统有两个步骤。第一步，是构建一个与宿主系统无关的新工具链（编译器、汇编器、链接器、库和一些有用的工具）。第二步则是使用该工具链，去构建其它的基础工具。

本章中编译得到的文件将被安装在目录 `$LFS/tools` 中，以确保在下一章中安装的文件和宿主系统生成的目录相互分离。由于此处编译的软件包都是临时性的，因此，我们不愿意它们污染后面即将构成的 LFS 系统。

## 5.2. 工具链技术说明

这一节解释总体构建方法之中的某些基本原理和技术细节。本节中的所有问题并无需马上消化。在进行实际构建的过程中，绝大部分的信息会变得愈加清晰。过程中随时都可以查阅本小节的内容。

纵览 Chapter#5 的目标是生成一个临时的系统，这个系统中包含一个已知的较好工具集，并且工具集可以独立于宿主系统。通过使用 `chroot`，其余各章中的命令将被包含在此环境中，以保证目标 LFS 系统能够洁净且无故障地生成。该构建过程的设计就是为了使得新读者承担最少的风险，同时还能有最好的指导价值。

### Note

在继续前，请留心工作平台的名称，它时常被称为目标三元组。要确定目标三元组的名称有一个简单的法子，那就是运行许多软件包的源码附带的脚本 `config.guess`。解压 `Binutils` 的源码并运行脚本：`./config.guess` 并注意它的输出。举一个例子，Intel 的 32 位处理器输出会是 `i686-pc-linux-gnu`。而在一个 64 位系统上，则会是 `x86_64-pc-linux-gnu`。

也请留心平台的动态链接器的名称，它时常被称为动态加载器（不要与 `Binutils` 中的标准链接器 `ld` 混为一谈）。动态链接器由 `Glibc` 提供，用于寻找和加载程序所需的共享库，为程序的运行做准备，以及运行程序。32 位的 Intel 机器，动态链接器的名称是 `ld-linux.so.2`（64 位系统则是 `ld-linux-x86-64.so.2`）。确定动态链接名的一个确定的方法，就是检查随机二进制文件，通过在宿主主机运行：`readelf -l <name of binary> | grep interpreter` 并注意它的输出。覆盖全平台的权威参考在 `Glibc` 源码树根目录的 `shlib-versions` 文件中。

下面是 Chapter#5 构建方法的几个关键技术点：

- 通过改变 `LFS_TGT` 变量的目标系统三段式中的「vendor（供应商）」字段，稍微调整一下工作平台的名称，以保证第一遍构建 `Binutils` 和 `GCC` 时能够生成兼容的交叉链接器和交叉编译器。此处的交叉链接器和交叉编译器生成的二进制文件与当前的硬件兼容，而不是用于其它的硬件架构。
- 临时库经交叉编译获得。由于交叉编译原本就不应该依赖于宿主系统，因此，通过降低宿主系统的头文件或库进入新工具的可能性，该方法可去除目标系统中潜在的污染。交叉编译的方式，还可以在 64 位硬件平台上同时构建出 32 位和 64 位库。
- 谨慎地操作 `GCC` 源码告诉编译器，哪个是即将被采用的目标动态链接器。

`Binutils` 是首个安装的包，因为无论是执行 `GCC` 还是 `Glibc` 的 `configure` 时，都将围绕关汇编器和链接器实施多项特性测试，来判断哪些软件特性要启用或是禁用。其重要性可能更甚于你对它的第一印象。`GCC` 或 `Glibc` 的错误配置会导致工具链出现难以捉摸的问题，可能直到整个构建过程接近尾声时，这些问题才会显现出来。不过，通常情况下，在你进行大量的无用功之前，一次测试套件的失败便会将该错误高亮出来。

`Binutils` 将其汇编器和链接器安装在两处，`/tools/bin` 和 `/tools/$LFS_TGT/bin`。有一处的工具是另一处的硬链接。链接器的一个重要方面是它的库搜索顺序。可以给 `ld` 传递参数 `--verbose` 获取详细信息。例如，`ld --verbose | grep SEARCH` 将说明当前的搜索路径及其顺序。通过编译一个 dummy（假）程序并向链接器传递 `--verbose` 参数，可显示 `ld` 都链接了哪些文件。例如，`gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded` 将显示链接过程中成功打开的所有文件。

下一个安装的包是 `GCC`。下面便是运行 `GCC` 的 `configure` 的输出示例：

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

基于上述原因，这很重要。这还说明了 GCC 的配置脚本并不会搜索 PATH 目录来寻找使用什么工具。不过，在 gcc 自身的实际运行中，并不需要使用同样的搜索路径。运行：`gcc -print-prog-name=ld` 可获知 gcc 使用是何种标准链接器。

在编译 dummy (假) 程序时，向 gcc 传递命令行选项 `-v` 可获得详细信息。例如，`gcc -v dummy.c` 将显示预处理器、编译和汇编阶段的详细信息，包括 gcc 的 include 搜索路径及其顺序。

下一个安装的是经过净化的 Linux API 头文件。这些头文件可使得标准 C 库 (Glibc) 与 Linux 内核提供的特性进行交互。

下一个安装的软件包是 Glibc。构建 Glibc 时，最重要的几个注意点是编译器、二进制工具和内核头文件。编译器通常不成问题，因为 Glibc 将一直使用传递给它配置脚本的，有关 `--host` 参数的编译器。如，在我们的场景中，编译器就是 `i686-lfs-linux-gnu-gcc`。而二进制工具和内核头文件可能就要复杂一些了。因此，请不要冒险，并利用可用的配置开关来强制正确的选择。configure 运行完毕，目录 `glibc-build` 下的文件 `config.make` 包含有所有的重要细节。需要注意的是，`CC="i686-lfs-gnu-gcc"` 用来控制使用哪个二进制工具，`-nostdinc` 和 `-isystem` 标志用来控制编译器的 include 搜索路径。这些条目强调了 Glibc 包的一个重要方面，即其构建机制是非常自给自足的，通常并不依赖默认工具链的默认设置。

在第二遍编译 Binutils 过程中，我们能够利用配置开关 `--with-lib-path` 来控制 ld 的库搜索路径。

第二遍编译 GCC 时，也需要修改其源代码，以告诉 GCC 使用新的动态链接器。如果不加修改，将导致 GCC 的自身程序中嵌入源自宿主系统目录 `/lib` 中的动态链接器，这违背了独立于宿主系统的目的。正是基于前面的这个出发点，核心工具链是自包含和自托管的。Chapter#5 其它的软件包都将在 `/tools` 中的新 Glibc 的基础上进行构建。

在进入 Chapter#6 中的 chroot 环境前，将安装的首个主要的软件包是 Glibc，这是因为它天生具有前面提及的自给自足特点。一旦将 Glibc 安装到 `/usr` 中，我们将快速改变工具链的默认设置，然后继续构建目标 LFS 系统的其余部分。

## 5.3. 通用编译指南

编译软件包时，本指南假设你已知晓这几点：

- 有几个软件包，在编译之前需要打补丁来规避一些问题。有的补丁在本章和下一章中都需要，但有些只有其中一章需要。因此，如果某章看起来缺少某个补丁的下载说明时，不用担心。安装补丁的时候也许会遇到关于 `offset` 或者 `fuzz` 的警告信息。别担心这些警告，补丁还是会成功安装的。
- 在大部分软件包的编译过程中，屏幕上都可能出现几个警告。这都很正常，可以安全地忽略。这些警告正如它们描述的那样，是对使用过时的 C 或 C++ 语法的警告，而不是这些语法不可用。C 语言的标准经常改变，一些软件包仍然在使用旧的标准。这并不是一个问题，不过确实会弹出警告。
- 最后确认一次，是否正确设置了 LFS 环境变量：

```
echo $LFS
```

确认输出显示的是 LFS 分区挂载点的路径，在我们的例子中，也就是 `/mnt/lfs`。

- 最后，有两个重要事项必须强调：

### Important

编译指南假定你已经正确地设置了 Host System Requirements 和符号链接：

- shell 使用的是 `bash`
- `sh` 是到 `bash` 的符号链接。
- `/usr/bin/awk` 是到 `gawk` 的符号链接。
- `/usr/bin/yacc` 是到 `bison` 的符号链接，或者是一个执行 `bison` 的小脚本。

## Important

再次强调构建的过程：

1. 把所有源文件和补丁放到 chroot 环境可访问的目录，例如 `/mnt/lfs/sources/`。但是千万不能把源文件放在 `/mnt/lfs/tools/` 中。
2. 进入到源文件目录。
3. 对于每个软件包：
  - a. 用 `tar` 程序解压要编译的软件包。在第五章中，确保解压软件包时你使用的是 `lfs` 用户。
  - b. 进入到解压后创建的目录中。
  - c. 根据指南说明编译软件包。
  - d. 回退到源文件目录。
  - e. 除非特别说明，删除解压出来的目录。

## 5.4. Binutils-2.32 - 第 1 遍

Binutils 软件包包含一个链接器、一个汇编器、以及其它处理目标文件的工具。

大致构建用时: 1 SBU  
所需磁盘空间: 580 MB

### 5.4.1. 安装交叉编译的 Binutils

#### Note

返回前面章节重新阅读注意事项。了解标记为重要的注意事项能在后面帮你省去很多麻烦。

第一个编译的软件包是 Binutils 软件包，这点很重要，因为 Glibc 和 GCC 都会对可用的链接器和汇编器执行各种测试，以决定启用它们自身的哪些功能。

Binutils 手册建议，在源码目录之外一个专门的编译目录里面编译 Binutils：

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

#### Note

为了衡量在本书中其余部分所使用 SBU 值，我们要测量一下这个软件包从配置到包括第一次安装在内的编译时间。为了轻松的做到这点，会用类似 `time { ./configure ... && ... && make install; }` 的方式将命令包裹在 time 命令中。

#### Note

第五章中，大致的构建 SBU 值和所需磁盘空间不包括测试套件数据。

现在准备编译 Binutils：

```
./configure --prefix=/tools \
            --with-sysroot=$LFS \
            --with-lib-path=/tools/lib \
            --target=$LFS_TGT \
            --disable-nls \
            --disable-werror
```

配置选项的含义：

`--prefix=/tools`

告诉配置脚本将 Binutils 程序安装到 /tools 文件夹。

`--with-sysroot=$LFS`

用于交叉编译，告诉编译系统在 \$LFS 中查找所需的目標系统库。

`--with-lib-path=/tools/lib`

指定需要配置使用的链接器的库路径。

`--target=$LFS_TGT`

因为 LFS\_TGT 变量中的机器描述和 config.guess 脚本返回的值略有不同，这个选项会告诉 configure 脚本调整 Binutils 的构建系统来构建一个交叉链接器。

`--disable-nls`

这会禁止国际化 (i18n)，因为国际化对临时工具来说没有必要。

`--disable-werror`

这会防止来自宿主编译器的警告事件导致停止编译。

继续编译软件包：

```
make
```

现在编译完成了。通常现在我们会运行测试套件，但在这个初期阶段，测试套件框架 (Tcl、Expect 和 DejaGNU) 还没有就绪。在此进行测试的收效甚微，因为第一遍编译的程序很快会被第二遍的代替。

如果是在 x86\_64 上构建，创建符号链接，以确保工具链的完整性：

```
case $(uname -m) in
  x86_64) mkdir -v /tools/lib && ln -sv lib /tools/lib64 ;;
esac
```

安装软件包:

```
make install
```

该软件包的详细信息位于 Section#6.16.2, “Binutils 内容”。



## 5.5. GCC-8.2.0 - 第 1 遍

GCC 软件包包括 GNU 编译器集，其中有 C 和 C++ 的编译器。

大致构建用时: 11 SBU

所需磁盘空间: 2.9 GB

### 5.5.1. 安装交叉编译的 GCC

现在 GCC 需要 GMP、MPFR 和 MPC 软件包。在你的主机发行版中可能并不包括这些软件包，它们将和 GCC 一起编译。将每个解压软件包到 GCC 的目录下，并重命名解压后得到的目录，以便 GCC 编译过程中能自动使用这些软件：

#### Note

读者经常对本章节产生误解。过程与之前的章节（软件包构建说明）中提到的一样。首先从源目录中解压 gcc 的源码包，然后进入创建的目录中。接着才可以执行下面的指令。

```
tar -xf ../mpfr-4.0.2.tar.xz
mv -v mpfr-4.0.2 mpfr
tar -xf ../gmp-6.1.2.tar.xz
mv -v gmp-6.1.2 gmp
tar -xf ../mpc-1.1.0.tar.gz
mv -v mpc-1.1.0 mpc
```

下面的指令将会修改 GCC 默认的动态链接器的位置，安装到 /tools 目录中的。并将 /usr/include 从 GCC 的 include 检索路径中移除。执行：

```
for file in gcc/config/{linux,i386/linux{,64}}.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_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

如果上面的内容看起来有些难以理解，那让我们慢慢消化吧。首先，我们复制文件 gcc/config/linux.h, gcc/config/i386/linux.h, 和 gcc/config/i386/linux64.h, 给复制的文件加上「.orig」后缀。然后第一个 sed 表达式在每个「/lib/ld」, 「/lib64/ld」或者「/lib32/ld」实例前面增加「/tools」, 第二个 sed 表达式替换「/usr」的硬编码实例。然后，我们添加这改变默认 startfile 前缀到文件末尾的定义语句。注意「/tools/lib/」后面的「/」是必须的。最后，我们用 touch 更新复制文件的时间戳。当与 cp -u 一起使用时，可以防止命令被无意中运行两次造成对原始文件意外的更改。

最后，在 x86\_64 的主机上，为 64 位的库设置默认目录名至「lib」：

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
    ;;
esac
```

GCC 手册建议在源目录之外一个专门的编译目录中编译 GCC：

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

准备编译 GCC:

```
../configure \
--target=$LFS_TGT \
--prefix=/tools \
--with-glibc-version=2.11 \
--with-sysroot=$LFS \
--with-newlib \
--without-headers \
--with-local-prefix=/tools \
--with-native-system-header-dir=/tools/include \
--disable-nls \
--disable-shared \
--disable-multilib \
--disable-decimal-float \
--disable-threads \
--disable-libatomic \
--disable-libgomp \
--disable-libmpx \
--disable-libquadmath \
--disable-libssp \
--disable-libvtv \
--disable-libstdcxx \
--enable-languages=c,c++
```

配置选项的含义:

`--with-newlib`

由于还没有可用的 C 库，这确保编译 libgcc 时定义了常数 `inhibit_libc`。这可以防止编译任何需要 libc 支持的代码。

`--without-headers`

创建一个完成的交叉编译器的时候，GCC 要求标准头文件和目标系统兼容。对于我们的目的来说，不需要这些头文件。这个选项可以防止 GCC 查找它们。

`--with-local-prefix=/tools`

GCC 会查找本地已安装的 include 文件的系统位置。默认是 `/usr/local`。把它设置为 `/tools` 能把主机位置中的 `/usr/local` 从 GCC 的搜索路径中排除。

`--with-native-system-header-dir=/tools/include`

GCC 默认会在 `/usr/include` 中查找系统头文件。和 `sysroot` 选项一起使用，会转换为 `$LFS/usr/include`。在后面两个章节中头文件会被安装到 `$LFS/tools/include`。这个选项确保 gcc 能正确找到它们。第二次编译 GCC 时，同样的选项可以保证不会去寻找主机系统的头文件。

`--disable-shared`

这个选项强制 GCC 静态链接到它的内部库。我们这样做是为了避免与主机系统可能出现的问题。

`--disable-decimal-float`, `--disable-threads`, `--disable-libatomic`, `--disable-libgomp`, `--disable-libmpx`, `--disable-libquadmath`, `--disable-libssp`, `--disable-libvtv`, `--disable-libstdcxx`

这些选项取消了对十进制浮点数扩展、线程

化、libatomic、libgomp、libmpx、libitm、libquadmath、libsanitizer、libssp、libvtv、libcilkrts 和 C++ 标准库的支持。这些功能在编译交叉编译器的时候会导致编译失败，对于交叉编译临时 libc 来说也没有必要。

`--disable-multilib`

在 x86\_64 机器上，LFS 还不支持 multilib 配置。这个选项对 x86 来说无害。

`--enable-languages=c,c++`

这个选项确保只编译 C 和 C++ 编译器。这些是现在唯一需要的语言。

运行命令编译 GCC:

```
make
```

现在编译完成了。在这里，通常会运行测试套件，但正如前面提到的，测试套件框架还没有准备好。在此进行测试的并没有太多好处，因为第一遍编译的程序很快会被取代。

安装软件包：

```
make install
```

该软件包的详细信息请参见：Section#6.21.2, “GCC 软件包内容”。



## 5.6. Linux-4.20.12 API 头文件

Linux API 头文件（在 linux-4.20.12.tar.xz 里）会将内核 API 导出给 Glibc 使用。

大致构建用时:               0.1 SBU  
所需磁盘空间:               937 MB

### 5.6.1. 安装 Linux API 头文件

Linux 内核需要展示供系统 C 库（在 LFS 中是 Glibc）使用的应用程序编程接口（API）。这通过在 Linux 内核源代码 tar 包中包括一些 C 头文件来完成。

确认这里没有陈旧的文件且不依赖于之前的操作：

```
make mrproper
```

从源代码中提取用户可见的内核头文件。把他们保存在一个临时本地文件夹中然后复制到所需的位置，因为解压过程会移除目标文件夹中任何已有的文件。

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

该软件包的详细信息请参见：Section#6.7.2, “Linux API 头文件内容”。

## 5.7. Glibc-2.29

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.

大致构建用时: 5.1 SBU

所需磁盘空间: 885 MB

### 5.7.1. Installation of Glibc

The Glibc documentation recommends building Glibc in a dedicated build directory:

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

Next, prepare Glibc for compilation:

```
../configure \
  --prefix=/tools \
  --host=$LFS_TGT \
  --build=$(../scripts/config.guess) \
  --enable-kernel=3.2 \
  --with-headers=/tools/include
```

The meaning of the configure options:

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`

The combined effect of these switches is that Glibc's build system configures itself to cross-compile, using the cross-linker and cross-compiler in `/tools`.

`--enable-kernel=3.2`

This tells Glibc to compile the library with support for 3.2 and later Linux kernels. Workarounds for older kernels are not enabled.

`--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.

During this stage the following warning might appear:

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

The missing or incompatible msgfmt program is generally harmless. This msgfmt program is part of the Gettext package which the host distribution should provide.

#### Note

There have been reports that this package may fail when building as a "parallel make". If this occurs, rerun the make command with a "-j1" option.

Compile the package:

```
make
```

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 'int main(){}' > dummy.c
$LFS_TGT-gcc 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/lib64/ld-linux-x86-64.so.2]
```

Note that for 32-bit machines, the interpreter name will be `/tools/lib/ld-linux.so.2`.

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.

Once all is well, clean up the test files:

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

## Note

Building Binutils in the section after next will serve as an additional check that the toolchain has been built properly. If Binutils fails to build, it is an indication that something has gone wrong with the previous Binutils, GCC, or Glibc installations.

Details on this package are located in Section#6.9.3, “Contents of Glibc.”

## 5.8. GCC-8.2.0 中的 Libstdc++

Libstdc++ 是标准的 C++ 库。需要用它来编译 C++ 代码 (GCC 的一部分是用 C++ 写的)，但是在构建 gcc-第 1 遍 时，我们需要推迟它的安装进程，因为依赖的 glibc，还未部署在 /tools 目录中。

Libstdc++ 是标准的 C++ 库。g++ 编译器正确运行需要它。

大致构建用时: 0.5 SBU

所需磁盘空间: 803 MB

### 5.8.1. 安装目标 Libstdc++

#### Note

Libstdc++ 是 GCC 源文件的一部分。你首先应该解压 GCC 的压缩包，然后进入 gcc-8.2.0 文件夹。

为 Libstdc++ 另外创建一个用于构建的文件夹并进入该文件夹：

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

准备编译 Libstdc++：

```
../libstdc++-v3/configure \
  --host=$LFS_TGT          \
  --prefix=/tools           \
  --disable-multilib        \
  --disable-nls             \
  --disable-libstdc++-threads \
  --disable-libstdc++-pch   \
  --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/8.2.0
```

配置选项的含义：

`--host=...`

指示使用我们刚才编译的交叉编译器，而不是 /usr/bin 中的。

`--disable-libstdc++-threads`

由于我们还没有编译 C 线程库，C++ 的也还不能编译。

`--disable-libstdc++-pch`

此选项防止安装预编译文件，此步骤并不需要。

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/8.2.0`

这是 C++ 编译器搜索标准 include 文件的位置。在一般的编译中，这个信息自动从顶层文件夹中传入 Libstdc++ configure 选项。在我们的例子中，必须明确给出这信息。

编译 libstdc++：

```
make
```

安装库：

```
make install
```

该软件包的详细信息请参见：Section#6.21.2, “GCC 软件包内容”。

## 5.9. Binutils-2.32 - 第 2 遍

Binutils 软件包包含一个链接器、一个汇编器、以及其它处理目标文件的工具。

大致构建用时: 1.1 SBU  
所需磁盘空间: 598 MB

### 5.9.1. 安装 Binutils

再次新建一个单独的编译文件夹：

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

准备编译 Binutils：

```
CC=$LFS_TGT-gcc          \
AR=$LFS_TGT-ar           \
RANLIB=$LFS_TGT-ranlib   \
../configure             \
  --prefix=/tools        \
  --disable-nls           \
  --disable-werror        \
  --with-lib-path=/tools/lib \
  --with-sysroot
```

新配置选项的含义：

```
CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib
```

因为这是真正的原生编译 Binutils，设置这些变量能确保编译系统使用交叉编译器和相关的工具，而不是宿主系统中已有的。

```
--with-lib-path=/tools/lib
```

这告诉配置脚本在编译 Binutils 的时候指定库搜索目录，此处将 `/tools/lib` 传递到链接器。

```
--with-sysroot
```

sysroot 功能使链接器可以找到包括在其命令行中的其它共享对象明确需要的共享对象。否则的话，在某些主机上一些软件包可能会编译不成功。

编译软件包：

```
make
```

安装软件包：

```
make install
```

现在，为下一章的「Re-adjusting」阶段准备链接器：

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

make 参数的含义：

```
-C ld clean
```

告诉 make 程序移除所有 ld 子目录中编译过的文件。

```
-C ld LIB_PATH=/usr/lib:/lib
```

这个选项重新编译 ld 子目录中的所有文件。在命令行中指定 Makefile 的 LIB\_PATH 变量可以使我们能够重写临时工具的默认值并指向正确的最终路径。该变量的值指定链接器的默认库搜索路径。下一章中会用到这个准备。

该软件包的详细信息请参见：Section#6.16.2, “Binutils 内容”。

## 5.10. GCC-8.2.0 - 第 2 遍

GCC 软件包包括 GNU 编译器集，其中有 C 和 C++ 的编译器。

大致构建用时: 14 SBU

所需磁盘空间: 3.4 GB

### 5.10.1. 安装 GCC

我们第一次编译 GCC 的时候安装了一些内部系统头文件。其中的一个 `limits.h` 会反过来包括对应的系统头文件 `limits.h`，在我们的例子中，是 `/tools/include/limits.h`。但是，第一次编译 gcc 的时候 `/tools/include/limits.h` 并不存在，因此 GCC 安装的内部头文件只是部分的自包含文件，并不包括系统头文件的扩展功能。这足以编译临时 `libc`，但是这次编译 GCC 要求完整的内部头文件。使用和正常情况下 GCC 编译系统使用的相同的命令创建一个完整版本的内部头文件：

```
cat gcc/limitx.h gcc/glimits.h gcc/limity.h > \
`dirname ${LFS_TGT-gcc -print-libgcc-file-name}`/include-fixed/limits.h
```

再一次更改 GCC 的默认动态链接器的位置，使用安装在 `/tools` 的那个。

```
for file in gcc/config/{linux,i386/linux{,64}}.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_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

如果是在 `x86_64` 环境上构建，为 64 位库改变默认目录名至「lib」：

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
    ;;
esac
```

和第一次编译 GCC 一样，它要求 GMP、MPFR 和 MPC 软件包。解压 tar 包并把它们重名为到所需的文件夹名称：

```
tar -xf ../mpfr-4.0.2.tar.xz
mv -v mpfr-4.0.2 mpfr
tar -xf ../gmp-6.1.2.tar.xz
mv -v gmp-6.1.2 gmp
tar -xf ../mpc-1.1.0.tar.gz
mv -v mpc-1.1.0 mpc
```

再次创建独立的编译文件夹：

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

在开始编译 GCC 之前，记住取消所有会覆盖默认优化选项的环境变量。

准备编译 GCC:

```
CC=$LFS_TGT-gcc \
CXX=$LFS_TGT-g++ \
AR=$LFS_TGT-ar \
RANLIB=$LFS_TGT-ranlib \
../configure \
  --prefix=/tools \
  --with-local-prefix=/tools \
  --with-native-system-header-dir=/tools/include \
  --enable-languages=c,c++ \
  --disable-libstdcxx-pch \
  --disable-multilib \
  --disable-bootstrap \
  --disable-libgomp
```

新配置选项的含义:

```
--enable-languages=c,c++
```

这个选项确保编译了 C 和 C++ 编译器。

```
--disable-libstdcxx-pch
```

不为 libstdc++ 编译预编译的头文件(PCH)。这会花费很多时间，却对我们没有用处。

```
--disable-bootstrap
```

对于原生编译的 GCC，默认是做一个「引导」构建。这不仅要编译 GCC 一次，而是会编译很多次。使用初次编译的程序去编译其自身第二次，然后同样地进行第三次。比较第二次和第三次迭代确保其能完美地复制自身。这也能预示编译是正确地。但是，LFS 的构建方法能够提供一个稳定的编译器，避免每次都需要重新引导。

编译软件包:

```
make
```

安装软件包:

```
make install
```

作为画龙点睛，这里创建一个符号链接。很多程序和脚本执行 `cc` 而不是 `gcc` 来保证程序的通用性，并且在所有的 Unix 类型的系统上都能用，而非仅局限于安装了 GCC 的 Unix 类型的系统。运行 `cc` 使得系统管理员不用考虑要安装那种 C 编译器：

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

## Caution

到了这里，必须停下来确认新工具链的基本功能（编译和链接）都是像预期的那样正常工作。运行下面的命令进行全面的检查：

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

如果一切工作正常的话，这里应该没有错误，最后一个命令的输出形式会是：

```
[Requesting program interpreter: /tools/lib64/ld-linux-x86-64.so.2]
```

注意 32 位机器的动态链接是 `/tools/lib/ld-linux.so.2`。

如果输出不是像上面那样或者根本就没有输出，那么可能某些地方出错了。调查并回溯这些步骤，找出问题所在并改正它。在继续之前必须解决这个问题。首先，使用 `gcc` 而不是 `cc` 再次进行全面的检查。如果能运行，就符号链接 `/tools/bin/cc` 就不见了。像上面介绍的那样新建符号链接。下一步，确认 `PATH` 是正确的。这能通过运行 `echo $PATH` 检验，验证 `/tools/bin` 在列表的前面。如果 `PATH` 是错误的，这意味着你可能不是以 `lfs` 用户的身份登录或者在前面 Section#4.4, “设置环境”中某些地方出现了错误。

一旦一切都顺利，清理测试文件：

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

该软件包的详细信息请参见：Section#6.21.2, “GCC 软件包内容”。

## 5.11. Tcl-8.6.9

Tcl 软件包包含工具命令语言 (Tool Command Language) 相关程序。

大致构建用时: 0.9 SBU  
所需磁盘空间: 66 MB

### 5.11.1. 安装 Tcl

此软件包和后面两个包 (Expect 和 DejaGNU) 用来为 GCC 和 Binutils 还有其他的一些软件包的测试套件提供运行支持。仅仅为了测试目的而安装三个软件包, 看上去有点奢侈, 虽然因为大部分重要的工具都能正常工作而并不需要做测试。尽管在本章中并没有执行测试套件 (并不做要求), 但是在 Chapter#6 中都要求执行这些软件包自带的测试套件。

注意, 这里的 Tcl 软件包用的是最小化安装的版本, 仅仅是为了运行 LFS 测试。需要完整版的软件包, 可参考 BLFS 的 Tcl 流程。

配置 Tcl 准备编译:

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

编译软件包:

```
make
```

现在编译已经完成。之前说过, 不强求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Tcl 的话可以用下面的命令:

```
TZ=UTC make test
```

Tcl 测试套件在宿主机某些特定条件下会失败, 原因很难推测。不过测试套件失败并不奇怪, 也不是什么严重的错误。参数 `TZ=UTC` 设定了时区和相应的世界标准时间 (UTC), 但是只在测试套件运行期间才有效。这个可以保证时钟测试能正常运行。关于 `TZ` 环境变量的细节请参阅本手册 第 7 章。

安装软件包:

```
make install
```

让安装的库文件可写, 这样之后可以删除调试符号。

```
chmod -v u+w /tools/lib/libtcl8.6.so
```

安装 Tcl 的头文件。后面的软件包 Expect 在编译的时候会用到。

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

现在创建几个必要的软链接:

```
ln -sv tclsh8.6 /tools/bin/tclsh
```

### 5.11.2. Tcl 软件包内容

安装的程序: tclsh (link to tclsh8.6) and tclsh8.6  
安装的库: libtcl8.6.so, libtclstub8.6.a

#### 简要介绍

tclsh8.6	Tcl 命令终端
tclsh	软链接到 tclsh8.6
libtcl8.6.so	Tcl 库
libtclstub8.6.a	Tcl Stub 库



## 5.12. Expect-5.45.4

Expect 软件包包含一个实现用脚本和其他交互式程序进行对话的程序。

大致构建用时: 0.1 SBU  
所需磁盘空间: 3.9 MB

### 5.12.1. 安装 Expect

首先，强制 Expect 的 configure 配置脚本使用 `/bin/stty` 替代宿主机系统里可能存在的 `/usr/local/bin/stty`。这样可以保证我们的测试套件工具在工具链的最后一次构建能够正常。

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

现在配置 Expect 准备编译：

```
./configure --prefix=/tools \
            --with-tcl=/tools/lib \
            --with-tclinclude=/tools/include
```

配置脚本参数的含义：

`--with-tcl=/tools/lib`

这个选项可以保证 configure 配置脚本会从临时工具目录里找 Tcl 的安装位置，而不是在宿主机系统中寻找。

`--with-tclinclude=/tools/include`

这个选项会给 Expect 显式地指定 Tcl 内部头文件的位置。通过这个选项可以避免 configure 脚本不能自动发现 Tcl 头文件位置的情况。

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Expect 的话可以用下面的命令：

```
make test
```

请注意 Expect 测试套件已知在某些宿主机特定情况下有过失败的情况，我们还没有完全把握。不过，在这里测试套件运行失败并不奇怪，也不认为是关键问题。

安装软件包：

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

make 参数的含义：

`SCRIPTS=""`

这个变量可以避免安装额外的、没有需求的 Expect 脚本。

### 5.12.2. Expect 软件包内容

安装的程序: expect  
安装的库: libexpect-5.45.so

#### 简要介绍

expect 基于脚本和其他交互式程序通信。

libexpect-5.45.so 包含一些函数允许 Expect 用作 Tcl 扩展或直接用于 C/C++（不用 Tcl）。

## 5.13. DejaGNU-1.6.2

DejaGNU 软件包包含了测试其他程序的框架。

大致构建用时: 少于 0.1 SBU

所需磁盘空间: 3.2 MB

### 5.13.1. 安装 DejaGNU

配置 DejaGNU 准备编译：

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

编译安装软件包：

```
make install
```

要测试编译结果，执行：

```
make check
```

### 5.13.2. DejaGNU 软件包内容

安装的程序: runtest

#### 简要介绍

runtest 一个封装脚本用于定位合适的 expect 终端然后执行 DejaGNU。

## 5.14. M4-1.4.18

M4 软件包包含一个宏处理器。

大致构建用时: 0.2 SBU

所需磁盘空间: 20 MB

### 5.14.1. 安装 M4

首先，对应 glibc-2.28 的需求做一些修复：

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

配置 M4 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 M4 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.14.2, “M4 软件包内容”。

## 5.15. Ncurses-6.1

Ncurses 软件包包含用于不依赖于特定终端的字符屏幕处理的库。

大致构建用时: 0.6 SBU  
所需磁盘空间: 41 MB

### 5.15.1. Installation of Ncurses

First, ensure that gawk is found first during configuration:

```
sed -i s/mawk// configure
```

Prepare Ncurses for compilation:

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

*--enable-widex*

This switch causes wide-character libraries (e.g., `libncursesw.so.6.1`) to be built instead of normal ones (e.g., `libncurses.so.6.1`). 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
ln -s libncursesw.so /tools/lib/libncurses.so
```

Details on this package are located in Section#6.24.2, “Ncurses 软件包内容.”

## 5.16. Bash-5.0

The Bash package contains the Bourne-Again SHell.

大致构建用时: 0.4 SBU  
所需磁盘空间: 67 MB

### 5.16.1. 安装 Bash

配置 Bash 准备编译：

```
./configure --prefix=/tools --without-bash-malloc
```

配置脚本参数的含义：

*--without-bash-malloc*

这个选项会禁用 Bash 的内存分配功能 (malloc)，这个功能已知会导致段错误。而禁用这个功能后，Bash 将使用 Glibc 的 malloc 函数，这样会更稳定。

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Bash 的话可以用下面的命令：

```
make tests
```

安装软件包：

```
make install
```

为使用 sh 终端的程序创建一个软链接：

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

关于这个软件包的详细资料请参见：Section#6.34.2, “Contents of Bash”。

## 5.17. Bison-3.3.2

The Bison package contains a parser generator.

大致构建用时: 0.3 SBU  
所需磁盘空间: 37 MB

### 5.17.1. 安装 Bison

配置 Bison 准备编译：

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

编译软件包：

```
make
```

为了测试结果，输入：

```
make check
```

安装软件包：

```
make install
```

该软件包的详细信息参见：Section#6.31.2, “Contents of Bison”。

## 5.18. Bzip2-1.0.6

Bzip2 软件包包含压缩和解压缩的程序。用 bzip2 压缩文本文件能获得比传统的 gzip 更好的压缩比。

大致构建用时：        少于 0.1 SBU

所需磁盘空间：        5.5 MB

### 5.18.1. 安装 Bzip2

Bzip2 软件包里没有 configure 配置脚本。用下面的命令编译和测试：

```
make
```

安装软件包：

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

关于这个软件包的详细资料请参见：Section#6.22.2, “Bzip2 软件包内容”。

## 5.19. Coreutils-8.30

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

大致构建用时: 0.8 SBU  
所需磁盘空间: 148 MB

### 5.19.1. 安装 Coreutils

配置 Coreutils 准备编译：

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

配置脚本参数的含义：

`--enable-install-program=hostname`

这个选项会允许构建和安装 `hostname` 程序——默认是不安装的，但是 Perl 测试套件需要它。

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Coreutils 的话可以用下面的命令：

```
make RUN_EXPENSIVE_TESTS=yes check
```

参数 `RUN_EXPENSIVE_TESTS=yes` 会告诉测试套件额外运行对某些系统开销相对大一些（主要是 CPU 运算能力和内存消耗）的测试用例，但是通常对 Linux 来说不是问题。

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.54.2, “Contents of Coreutils”。



## 5.20. Diffutils-3.7

Diffutils 软件包包含显示文件和目录差异的程序。

大致构建用时: 0.2 SBU

所需磁盘空间: 26 MB

### 5.20.1. 安装 Diffutils

配置 Diffutils 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Diffutils 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.56.2, “Diffutils 软件包内容”。

## 5.21. File-5.36

File 软件包包括一个判断给定的某个或某些文件文件类型的工具。

大致构建用时: 0.1 SBU

所需磁盘空间: 18 MB

### 5.21.1. 安装 File

配置 File 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 File 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.12.2, “File 软件包内容”。

## 5.22. Findutils-4.6.0

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).

大致构建用时: 0.3 SBU  
所需磁盘空间: 36 MB

### 5.22.1. 安装 Findutils

首先，对应 glibc-2.28 的需求做一些修复：

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' gl/lib/*.c
sed -i '/unistd/a #include <sys/sysmacros.h>' gl/lib/mountlist.c
echo "#define _IO_IN_BACKUP 0x100" >> gl/lib/stdio-impl.h
```

配置 Findutils 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Findutils 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.58.2, “Contents of Findutils”。

## 5.23. Gawk-4.2.1

The Gawk package contains programs for manipulating text files.

大致构建用时: 0.2 SBU  
所需磁盘空间: 43 MB

### 5.23.1. 安装 Gawk

配置 Gawk 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Gawk 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.57.2, “Contents of Gawk”。

## 5.24. Gettext-0.19.8.1

Gettext 软件包包含用于国际化和地区化的工具。这允许用 NLS (Native Language Support, 本地语言支持) 编译程序, 使得能以用户的本地语言输出信息。

大致构建用时: 0.9 SBU  
所需磁盘空间: 173 MB

### 5.24.1. 安装 Gettext

对于我们这次用到的临时工具集, 我们只需要编译安装 Gettext 软件包里的 3 个程序。

配置 Gettext 准备编译:

```
cd gettext-tools
EMACS="no" ./configure --prefix=/tools --disable-shared
```

配置脚本参数的含义:

*EMACS="no"*

这个选项会禁止配置脚本检测安装 Emacs Lisp 文件的位置, 已知在某些系统中会引起错误。

*--disable-shared*

这次我们不需要安装任何的 Gettext 动态库, 所以不需要编译。

编译软件包:

```
make -C gnulib-lib
make -C intl pluralx.c
make -C src msgfmt
make -C src msgmerge
make -C src xgettext
```

因为只编译了 3 个程序, 在不编译 Gettext 的额外支持库的情况下, 测试套件是不可能成功运行的。所以在这个阶段建议不要尝试运行测试套件。

安装 msgfmt、msgmerge 和 xgettext 程序:

```
cp -v src/{msgfmt,msgmerge,xgettext} /tools/bin
```

关于这个软件包的详细资料请参见: Section#6.47.2, “Gettext 软件包内容”

## 5.25. Grep-3.3

Grep 软件包包含用于在文件中搜索的程序。

大致构建用时: 0.2 SBU

所需磁盘空间: 24 MB

### 5.25.1. 安装 Grep

配置 Grep 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Grep 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.33.2, “Grep 软件包内容”。

## 5.26. Gzip-1.10

The Gzip package contains programs for compressing and decompressing files.

大致构建用时: 0.1 SBU

所需磁盘空间: 10 MB

### 5.26.1. Installation of Gzip

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#6.62.2, “Contents of Gzip.”

## 5.27. Make-4.2.1

The Make package contains a program for compiling packages.

大致构建用时: 0.1 SBU  
所需磁盘空间: 13 MB

### 5.27.1. Installation of Make

First, work around an error caused by glibc-2.27 and later:

```
sed -i '211,217 d; 219,229 d; 232 d' glob/glob.c
```

Prepare Make for compilation:

```
./configure --prefix=/tools --without-guile
```

The meaning of the configure option:

*--without-guile*

This ensures that Make-4.2.1 won't link against Guile libraries, which may be present on the host system, but won't be available within the chroot environment in the next chapter.

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#6.66.2, “Contents of Make.”



## 5.28. Patch-2.7.6

Patch 软件包包含一个通过打「补丁」创建或修改文件的程序，补丁文件通常由 diff 程序生成。

大致构建用时: 0.2 SBU  
所需磁盘空间: 12 MB

### 5.28.1. 安装 Patch

配置 Patch 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Patch 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.67.2, “Patch 软件包内容”。

## 5.29. Perl-5.28.1

The Perl package contains the Practical Extraction and Report Language.

大致构建用时: 1.6 SBU  
所需磁盘空间: 275 MB

### 5.29.1. 安装 Perl

配置 Perl 准备编译：

```
sh Configure -des -Dprefix=/tools -Dlibs=-lm -Uloclibpth -Ulocincpth
```

配置选项的含义：

*-des*

这是三个选项地的组合：-d 对所有选项使用默认值；-e 确保完成所有任务；-s 静默，不产生非必要输出。

*-Uloclibpth amd -Ulocincpth*

这些条目中未定义的变量，会促使配置过程去寻找宿主系统中已经存在的本地安装的组件。

编译软件包：

```
make
```

虽然 Perl 软件包自带测试套件，最好还是等下一章中它被完整安装之后再运行。

这次我们只需要安装一小部分的应用和库。

```
cp -v perl cpan/podlators/scripts/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.28.1
cp -Rv lib/* /tools/lib/perl5/5.28.1
```

关于这个软件包的详细资料请参见：Section#6.40.2, “Contents of Perl”。

## 5.30. Python-3.7.2

软件包 Python 3 包含了 Python 的开发环境。对于面向对象编程，书写脚本，构建大型程序的原型，或者开发整个应用程序而言，非常有用。

大致构建用时: 1.5 SBU  
所需磁盘空间: 371 MB

### 5.30.1. Installation of Python

This package first builds the Python interpreter, then some standard Python modules. The main script for building modules is written in Python, and uses hard-coded paths to the host `/usr/include` and `/usr/lib` directories. To prevent them from being used, issue:

```
sed -i '/def add_multiarch_paths/a \          return' setup.py
```

Prepare Python for compilation:

```
./configure --prefix=/tools --without-ensurepip
```

The meaning of the configure option:

*--without-ensurepip*

This switch disables the Python installer, which is not needed at this stage.

Compile the package:

```
make
```

Compilation is now complete. The test suite requires TK and and X Windows and cannot be run at this time.

Install the package:

```
make install
```

Details on this package are located in Section#6.51.2, “Python 3 软件包内容.”

## 5.31. Sed-4.7

Sed 软件包包含一个流编辑器。

大致构建用时: 0.2 SBU

所需磁盘空间: 20 MB

### 5.31.1. 安装 Sed

配置 Sed 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Sed 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.28.2, “Sed 软件包内容”。

## 5.32. Tar-1.31

The Tar package contains an archiving program.

大致构建用时: 0.3 SBU  
所需磁盘空间: 38 MB

### 5.32.1. 安装 Tar

配置 Tar 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Tar 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.69.2, “Contents of Tar”。

## 5.33. Texinfo-6.5

Texinfo 软件包包含用于读、写以及转换信息页的程序。

大致构建用时: 0.3 SBU  
所需磁盘空间: 104 MB

### 5.33.1. 安装 Texinfo

配置 Texinfo 准备编译：

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

#### Note

作为配制过程的一部分，有一个测试会指出 TestXS\_la-TestXS.lo 有一处错误。这与 LFS 没有关系，可以忽略。

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Texinfo 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.70.2, “Texinfo 软件包内容”。

## 5.34. Util-linux-2.33.1

The Util-linux package contains miscellaneous utility programs.

大致构建用时: 1 SBU  
所需磁盘空间: 147 MB

### 5.34.1. Installation of Util-linux

Prepare Util-linux for compilation:

```
./configure --prefix=/tools \
            --without-python \
            --disable-makeinstall-chown \
            --without-systemdsystemunitdir \
            --without-ncurses \
            PKG_CONFIG=""
```

The meaning of the configure option:

*--without-python*

This switch disables using Python if it is installed on the host system. It avoids trying to build unneeded bindings.

*--disable-makeinstall-chown*

This switch disables using the `chown` command during installation. This is not needed when installing into the `/tools` directory and avoids the necessity of installing as root.

*--without-ncurses*

This switch disables using the `ncurses` library for the build process. This is not needed when installing into the `/tools` directory and avoids problems on some host distros.

*--without-systemdsystemunitdir*

On systems that use `systemd`, the package tries to install a `systemd` specific file to a non-existent directory in `/tools`. This switch disables the unnecessary action.

`PKG_CONFIG=""`

Setting this environment variable prevents adding unneeded features that may be available on the host.

Note that the location shown for setting this environment variable is different from other LFS sections where variables are set preceding the command. This location is shown to demonstrate an alternative way of setting an environment variable when using `configure`.

Compile the package:

```
make
```

Install the package:

```
make install
```

Details on this package are located in Section#6.75.3, "Contents of Util-linux."

## 5.35. Xz-5.2.4

Xz 软件包包含用于压缩和解压文件的程序。它提供 lzma 和更新的 xz 压缩格式功能。和传统的 gzip 或 bzip2 命令相比，用 xz 压缩文本文件能获得更好的压缩率。

大致构建用时: 0.2 SBU  
所需磁盘空间: 18 MB

### 5.35.1. 安装 Xz

配置 Xz 准备编译：

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

编译软件包：

```
make
```

现在编译已经完成。之前说过，不要求为本章中所构建的临时工具运行测试套件。不过你仍然要测试 Xz 的话可以用下面的命令：

```
make check
```

安装软件包：

```
make install
```

关于这个软件包的详细资料请参见：Section#6.45.2, “Xz 软件包内容”。



## 5.36. 清理无用内容

本小节里的步骤是可选的，但如果你的 LFS 分区容量比较小，知道有些不必要的内容可以被删除也是挺好的。目前编译好的可执行文件和库大概会有 70MB 左右不需要的调试符号。可以通过下面的命令移除这些符号：

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

这两个命令会跳过一些文件，并提示不可识别的文件格式。大多数是脚本文件而不是二进制文件。同样还可以用宿主系统里的 `strip` 命令为 `/tools` 目录下的 `strip` 二进制文件清理无用内容。

注意不要对库文件使用 `--strip-unneeded` 选项。静态库会被损坏导致整个工具链将会需要全部重新编译。

更节省更多空间，还可以删除帮助文档：

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

删除不需要的文件：

```
find /tools/{lib,libexec} -name \*.la -delete
```

这个时候，你应该在 `$LFS` 分区中为下个阶段编译安装 Glibc 和 GCC 预留至少 3GB 剩余空间。如果你可以编译安装 Glibc，那其他的就不会有问题了。

## 5.37. 改变属主

### Note

本书余下部分的命令都必须以 `root` 用户身份执行而不再是 `lfs` 用户。另外，再次确认下 `$LFS` 变量在 `root` 用户环境下也有定义。

当前，`$LFS/tools` 目录属于 `lfs` 用户，这是一个只存在于宿主系统上的帐号。如果继续保持 `$LFS/tools` 目录的现状，其中的文件将属于一个没有相关联帐号的用户 ID。这很危险，因为随后创建的用户有可能会分配到相同的用户 ID，从而变成 `$LFS/tools` 目录及其中所有文件的属主，以致留下恶意操作这些文件的可能。

为了解决这个问题，你可以在随后新的 `lfs` 系统里创建 `/etc/passwd` 文件时增加一个 `lfs` 用户，并注意给它分配和宿主系统里相同的用户和组 ID。不过更好的方式是，通过下面的命令将 `$LFS/tools` 目录的属主改为 `root` 用户：

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

尽管 `$LFS/tools` 目录可以在 LFS 系统构建完成后删除，但仍然可以保留下来用于 构建额外的相同版本 LFS 系统。备份 `$LFS/tools` 目录到底有多少好处取决于你个人。

### Caution

如果你想保留临时工具用来构建新的 LFS 系统，现在就要备份好。本书随后第六章中的指令将对当前的工具做些调整，导致在构建新系统时会失效。

## Part III. 构建 LFS 系统

# Chapter 6. Installing Basic System Software

## 6.1. 简介

在本章中，我们会进入构建环境开始认真地构建 LFS 系统了。我们将 chroot 到之前准备好的临时迷你 Linux 系统，做一些最后的准备工作，然后就开始安装软件包。

安装软件很简单。尽管很多时候安装指令能更短而且更具通用性，但我们还是选择为每个软件包都提供完整的指令，以减小引起错误的可能性。了解 Linux 系统如何工作的关键就是知道每个软件包的作用以及为什么你（或系统）需要它。

我们不建议在编译时使用优化。这虽然可以让程序运行得快那么一点点，但是却也有可能增加编译难度，以及在运行时出问题。如果在打开优化后编译失败，请试一下关闭优化编译看看。就算打开优化通过了编译，考虑到源代码和编译工具之间的复杂交互，仍然存在编译不正确的风险。还有需要注意 `-march` 和 `-mtune` 选项除了本书指定的值都未经测试。这有可能导致工具链软件包 (Binutils、GCC 和 Glibc) 发生问题。对比使用编译优化带来的好处与风险，这样做经常是得不偿失。第一次构建 LFS 系统还是推荐不要使用自定义优化。这样构建出来的系统一样会运行得很快，于此同时还很稳定。

本章里安装软件包的顺序需要严格遵守，这是为了保证不会有程序意外地依赖与 `/tools` 路径的硬链接相关的目录。同样的理由，不要同时编译不同的软件包。并行地编译也许能节省一点时间（特别是在双 CPU 电脑上），但是它可能会导致程序里存在包含到 `/tools` 目录的硬链接，这样的话在这个目录移除后程序就不能正常工作了。

在安装指令之前，每个页面都提供了关于软件包的信息，包括其中所包含内容的精确描述，构建需求的大致时间，以及在过程中需求磁盘空间的大小。在安装指令之后，是一个该软件包即将安装的程序和库（及概要说明）的列表。

### Note

第六章里软件包的 SBU 数值和所需磁盘空间包含了可能存在的测试套件数据。

### 6.1.1. 关于库

总的来说，LFS 的编辑们并不推荐构建和安装静态库。许多静态库的初衷已经赶不上现在的 Linux 系统了。而且将静态库链接到程序还有不好之处。假设库更新需要移除一个安全问题，所有使用该静态库的程序都需要重新链接到新的库。由于静态库并不会总那么明显，有哪些相关的程序（以及需要链接的程序）很可能都不知道。

第六章的程序，我们移除或禁止了大部分静态库的安装。通常通过在 `configure` 命令中使用 `--disable-static` 项，便可以做到。有些情况下，可能用到其他代替的办法。当然也有少数情况，特别是 `glibc` 和 `gcc` 使用的静态库在软件包的构建过程中是必不可少的。

更多关于库的讨论，请参考 BLFS 中 库：静态还是共享？ 章节。

## 6.2. 准备虚拟内核文件系统

内核会挂载几个文件系统用于自己和用户空间程序交换信息。这些文件系统是虚拟的，并不占用实际磁盘空间，它们的内容会放在内存里。

开始先创建将用来挂载文件系统的目录：

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

### 6.2.1. 创建初始设备节点

在内核引导系统的时候，它依赖于几个设备节点，特别是 `console` 和 `null` 两个设备。设备点必须创建在硬盘上以保证在 `udev` 启动前是可用的，特别是在使用 `init=/bin/bash` 启动 Linux 时。运行以下命令创建设备节点：

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

## 6.2.2. 挂载和激活 /dev

通常激活 /dev 目录下设备的方式是在 /dev 目录挂载一个虚拟文件系统（比如 tmpfs），然后允许在检测到设备或打开设备时在这个虚拟文件系统里动态创建设备节点。这个通常是在启动过程中由 udev 完成。由于我们的新系统还没有 udev，也没有被引导，有必要手动挂载和激活 /dev 这可以通过绑定挂载宿主主机系统的 /dev 目录来实现。绑定挂载是一种特殊的挂载模式，它允许在另外的位置创建某个目录或挂载点的镜像。运行下面的命令来实现：

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

## 6.2.3. 挂载虚拟文件系统

现在挂载剩下的虚拟内核文件系统：

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

挂载选项 devpts 的含义：

*gid=5*

以确保所有由 devpts 创建的设备节点属于 GID 5 之下。这个 ID 日后将用于 tty 组。我们使用 GID 来代替组名，原因是宿主系统可能为 tty 组使用其他的 ID。

*mode=0620*

以确保所有 devpts 创建的设备节点的属性为 0620（属主用户可读写，组成员可写）。与上一个选项同时使用，可以保证 devpts 所创建的设备节点能满足 grantpt() 函数的要求，这就意味着不需要 Glibc（默认不安装的）帮助程序 pt\_chown 了。

在某些宿主主机系统里，/dev/shm 是一个指向 /run/shm 的软链接。这个 /run 下的 tmpfs 文件系统已经在之前挂载了，所以在这里只需要创建一个目录。

```
if [ -h $LFS/dev/shm ]; then
    mkdir -pv $LFS/$(readlink $LFS/dev/shm)
fi
```

## 6.3. 软件包管理

软件包管理经常被请求加入到 LFS 手册中。软件包管理器可以追踪安装的文件，方便软件包的移除和升级。不仅是二进制执行文件和库文件，包管理器还会处理配置文件的安装。在你想太多之前，答案是不——本节不讨论也不安利任何特定的软件包管理器，只是总结了一下关于软件包管理的常用技术和工作原理。对你而言最完美的软件包管理器可能就在这些技术之中，也可能由这些技术的其中几个组合而成。本节还简要的提了一些在升级软件包时可能遇到的问题。

为什么 LFS 或 BLFS 手册里不采用任何软件包管理器的一些原因：

- 使用软件包管理偏离了本手册的主要目标——教大家 Linux 系统是如何构建出来的。
- 存在很多软件包管理的解决方案，每一个都有自己的长处和缺点。很难选择一种适合所有人的方式。

关于软件包管理有很多资料，可以访问 Hints Project 看看是否可以解决你的需求。

### 6.3.1. 升级问题

软件包管理器可以在软件新版本发布后轻松升级。一般来说 LFS 和 BLFS 手册里的指令是可以用来升级版本。下面是一些在你准备升级软件包时需要注意的事情，特别是运行中系统需更加注意。

- 如果需要升级 Glibc 到新版本（比如，从 glibc-2.19 升级到 glibc-2.20），重新构建整个 LFS 会比较安全。虽然你也许能够按依赖关系重新编译所有的软件包，不过我们不建议这样做。
- 如果某个包含的动态库的软件包升级了，而且库名字有所改变，那么所有动态链接到这个库的软件包都需要重新链接新的库。（请注意软件包版本和库名字并不存在相关性）。举个例子，说软件包 foo-1.2.3 安装了一个叫 libfoo.so.1 的动态库。然后你将软件包升级到了新版本 foo-1.2.4 安装的动态库叫 libfoo.so.2 了。在这种情况下，所有动态链接到 libfoo.so.1 的软件包都需要重新编译链接到 libfoo.so.2。注意在所有依赖软件包重新编译完成之前，请勿删除之前的库文件。

## 6.3.2. 软件包管理技术

下面介绍一些常见的软件包管理技巧。在决定用哪种包管理方式之前，先研究一下各种不同的技术，特别是某些方案的不足之处。

### 6.3.2.1. 所有一切都在我脑袋里！

是的，这也算一种软件包管理技术。有些人觉得不需要管理软件包，是因为他们非常熟悉软件包，知道每个包都安装了哪些文件。也有些用户不需要管理软件包，是因为他们会在某个软件包有更改后重建整个系统。

### 6.3.2.2. 在独立目录里安装

这是一种简单的软件包管理方式，不需要其他额外的软件来管理软件的安装。每一个软件包都被装到一个独立的目录里。例如，软件包 `foo-1.1` 安装到目录 `/usr/pkg/foo-1.1` 中并创建一个软链接 `/usr/pkg/foo` 指向 `/usr/pkg/foo-1.1`。在安装新版本 `foo-1.2` 的时候，它会被装到目录 `/usr/pkg/foo-1.2` 中，然后用指向新版本的软链接替代之前的软链接。

类似 `PATH`、`LD_LIBRARY_PATH`、`MANPATH`、`INFOPATH` 和 `CPPFLAGS` 之类的环境变量变量需要包含 `/usr/pkg/foo` 目录。在管理大量软件包时，这种方式就不可行了。

### 6.3.2.3. 软链接方式软件包管理

这是前一种软件包管理技术的变种。每个软件包的安装方式都和之前的方式类似。但不是建立目录的软链接，而是把每个文件都链接到 `/usr` 目录结构里。这样就不需要扩展环境变量了。通过自动创建这些可由用户自行创建的软链，许多软件包管理器就采用了这种方式进行管理的。其中比较流行的有 `Stow`、`Epkg`、`Graft` 和 `Depot`。

这种安装方式需要伪装，这样软件包会认为自己被装到了 `/usr` 目录下，而实际上它被装到了 `/usr/pkg` 目录下。在这种方式下，安装并不是一件琐碎的小事。例如，假如你准备安装一个软件包 `libfoo-1.1`。下面的指令可能不会正确地安装：

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

安装本身倒是没有问题，但是可能一些依赖包不会像你期望的那样链接到 `libfoo` 库。如果你要编译一个链接 `libfoo` 的软件，你可能会注意到它实际上链接到的是 `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` 而不是你所期望的 `/usr/lib/libfoo.so.1`。正确的方式是使用 `DESTDIR` 策略来伪装软件包的安装过程。这种方式需要像下面这样操作：

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

大多数软件包支持这种方式，但也有一些例外。对于不兼容的软件包，你可能需要自己手动安装，或许你会发现将这些有问题的包安装到 `/opt` 目录下会更简单些。

### 6.3.2.4. 基于时间戳

在这种方式里，在安装之前会创建一个时间戳文件。在安装之后，用一行简单的 `find` 命令加上合适的参数就可以生成在时间戳文件创建之后所安装的所有文件列表。有一个采用这种方式的包管理器叫做 `install-log`。

这种方式的优点是非常简单，但是它有两个缺陷。比如，在安装过程中，所安装文件采用的是其它时间戳而不是当前时间，那这些文件将不能被软件包管理器跟踪到。还有，这种方式只能在一次安装一个软件包的情况下使用。如果在不同的终端里同时安装两个不同的软件包，此时的安装日志就不可靠了。

### 6.3.2.5. 追踪安装脚本

在这种方式里，安装脚本所使用的命令都会被记录下来。有两种技术，一种是：

设定环境变量 `LD_PRELOAD` 指向一个在安装前预加载的库。在安装过程中，这个库会追踪软件包安装脚本里所包含的各种执行文件比如 `cp`、`install`、`mv`，以及追踪会修改文件系统的系统调用。要让这种方式有效的，所有的执行文件需要动态链接到没有 `suid` 或 `sgid` 标志位的库。预加载这个库可能会引起安装过程中一些意外的副作用。因此，建议做一些测试以保证软件包管理器不会造成破坏并且记录了所有适当的文件。

第二种技术是使用 `strace` 命令，它会记录下安装脚本执行过程中所有的系统调用。

### 6.3.2.6. 创建软件包存档

在这种方式里，像之前的软链接软件包管理方式里所描述的那样，软件包被伪装安装到一个独立的目录树里。在安装完成后，会将已安装文件打包成一个软件包存档。然后这个存档会用来在本地机器或其他机器上安装软件包。

这种方式为商业发行版中的大多数包管理器所采用。例子有 RPM（顺带提一下，这也是 Linux 标准规范 中指定的包管理器），pkg-utils，Debian 的 apt，和 Gentoo 的 Portage 系统。如何在 LFS 系统里采用这种包管理方式的简单描述，请参看 <http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt>。

创建带有依赖关系的软件包存档非常复杂，已经超出 LFS 手册范围了。

Slackware 使用一个基于 tar 的系统来创建软件包存档。这套系统不像那些更复杂的包管理器，有意地不处理包依赖关系。关于 Slackware 包管理器的详细信息，请参看 <http://www.slackbook.org/html/package-management.html>。

### 6.3.2.7. 基于用户的软件包管理

这种方式，是 LFS 特有的，由 Matthias Benkmann 所设计，可以在 Hints Project 中找到。在这种方式里，每个软件包都由一个单独的用户安装到标准的位置。文件属于某个软件包可以通过检查用户 ID 轻松识别出来。关于这种方式的利弊比较复杂，就不再本节中赘述了。详细的信息请参看 [http://www.linuxfromscratch.org/hints/downloads/files/more\\_control\\_and\\_pkg\\_man.txt](http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt)。

### 6.3.3. 在多个系统上部署 LFS

LFS 系统的一个优点是没有那种需要依赖其在磁盘系统中的位置的文件。克隆一份 LFS 到和宿主机相似配置的机器上，简单到只要对包含根目录的 LFS 分区（对于一个基本的 LFS 构建不压缩的话大概有 250MB）使用 tar 命令打包，然后通过网络传输或光盘拷贝到新机器上展开即可。在这之后，需要调整一些配置文件。需要更新的配置文件包括：/etc/hosts，/etc/fstab，/etc/passwd，/etc/group，/etc/shadow，和 /etc/ld.so.conf。

根据系统硬件和原始内核配置文件的差异，可能还需要重新编译一下内核。

#### Note

据报告，当这样的复制发生在两个相近却又不完全相同的架构时会发生问题。例如，Intel 系统的指令集就和 AMD 处理器的不同，还有一些较新版的处理器可能会有一些在较早版本中不能支持的指令。

最后，通过 Section#8.4, “Using GRUB to Set Up the Boot Process”中介绍的方法来使新系统可以引导。

## 6.4. 进入 Chroot 环境

现在可以切换到 chroot 环境开始构建和安装最终的 LFS 系统了。以 root 用户运行下面的命令进入此环境，从现在开始，就只剩下准备的那些临时工具了：

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

给 env 命令传递 -i 选项会清除这个 chroot 切换进去的环境中的所有变量。随后，只需重新设定 HOME、TERM、PS1、和 PATH 变量。TERM=\$TERM 将会把 TERM 设定成 chroot 外环境相同的值。许多程序需要这个变量才能正常工作，比如 vim 和 less。如果还需设定其他变量，如 CFLAGS 或 CXXFLAGS，正好在这一起设置了。

在这之后，LFS 变量就不再需要了，因为后面所有工作都将被限定在 LFS 文件系统中。因为我们已经告诉 Bash 终端 \$LFS 就是当前的根目录 (/) 目录。

注意要将 /tools/bin 放在 PATH 变量的最后。意思是在每个软件的最后版本编译安装好后就不再使用临时工具了。这还需要让 shell 不要「记住」每个可执行文件的位置——这样的话，还要给 bash 加上 +h 选项来关闭其哈希功能。

注意一下 bash 的提示符是 `I have no name!` 这是正常的，因为这个时候 `/etc/passwd` 文件还没有被创建。

## Note

非常重要，从本章开始，后续章节中的命令都要在 `chroot` 环境下运行。如果因为某种原因（比如说重启）离开了这个环境，请保证要按照 Section#6.2.2, “挂载和激活 `/dev`”和 Section#6.2.3, “挂载虚拟文件系统”中所说的那样挂载虚拟内核文件系统，并在继续构建之前重新运行 `chroot` 进入环境。

## 6.5. 创建目录

现在准备创建 LFS 文件系统里的一些目录结构。使用下面的命令创建一个标准的目录树：

```
mkdir -pv /{bin,boot,etc}/{opt,sysconfig},home,lib/firmware,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,svr,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/{color,dict,doc,info,locale,man}
mkdir -v /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -v /usr/libexec
mkdir -pv /usr/{,local/}share/man/man{1..8}

case $(uname -m) in
  x86_64) mkdir -v /lib64 ;;
esac

mkdir -v /var/{log,mail,spool}
ln -sv /run /var/run
ln -sv /run/lock /var/lock
mkdir -pv /var/{opt,cache,lib/{color,misc,locate},local}
```

一般目录默认会按 755 的权限创建，但是这并不适用于所有的目录。在上面的命令里，有两个改动——一个是 `root` 的 `home` 目录，另一个是存放临时文件的目录。

第一个模式改动能保证不是所有人都能进入 `/root` 目录——同样一般用户也需要为他/她的 `home` 目录设置这样的模式。第二个模式改动能保证所有用户对目录 `/tmp` 和 `/var/tmp` 都是可写的，但又不能移除其他用户的文件。后面的这个限制是由所谓的「粘滞位」实现的，即位掩码 1777 中的最高位（1）。

### 6.5.1. 关于 FHS 兼容性

这个目录树是基于文件系统目录结构标准（FHS）（参看 <https://wiki.linuxfoundation.org/en/FHS>）。FHS 标准还规定了要有 `/usr/local/games` 和 `/usr/share/games` 目录。我们只创建了我们需要的目录。然而，如果你更喜欢严格遵守 FHS 标准，创建这些目录也无妨。

## 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,cats,chmod,dd,echo,ln,mkdir,pwd,rm,stat,touch} /bin
ln -sv /tools/bin/{env,install,perl,printf} /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{,.6} /usr/lib

install -vdm755 /usr/lib/pkgconfig

ln -sv bash /bin/sh
```

The purpose of each link:

`/bin/bash`

Many bash scripts specify `/bin/bash`.



`/bin/cat`

This pathname is hard-coded into Glibc's configure script.

`/bin/dd`

The path to `dd` will be hard-coded into the `/usr/bin/libtool` utility.

`/bin/echo`

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

`/usr/bin/env`

This pathname is hard-coded into some packages build procedures.

`/usr/bin/install`

The path to `install` will be hard-coded into the `/usr/lib/bash/Makefile.inc` file.

`/bin/ln`

The path to `ln` will be hard-coded into the `/usr/lib/perl5/5.28.1/<target-triplet>/Config_heavy.pl` file.

`/bin/pwd`

Some configure scripts, particularly Glibc's, have this pathname hard-coded.

`/bin/rm`

The path to `rm` will be hard-coded into the `/usr/lib/perl5/5.28.1/<target-triplet>/Config_heavy.pl` file.

`/bin/stty`

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

`/usr/bin/perl`

Many Perl scripts hard-code this path to the perl program.

`/usr/lib/libgcc_s.so{,.1}`

Glibc needs this for the pthreads library to work.

`/usr/lib/libstdc++.so{,.6}`

This is needed by several tests in Glibc's test suite, as well as for C++ support in GMP.

`/bin/sh`

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

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

```
ln -sv /proc/self/mounts /etc/mtab
```

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

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

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/var/run/dbus:/bin/false
systemd-bus-proxy:x:72:72:systemd Bus Proxy:/:/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/:/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/:/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/:/bin/false
systemd-network:x:76:76:systemd Network Management:/:/bin/false
systemd-resolve:x:77:77:systemd Resolver:/:/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/:/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/:/bin/false
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:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
kvm:x:61:
systemd-bus-proxy:x:72:
systemd-journal-gateway:x:73:
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
wheel:x:97:
nogroup:x:99:
users:x:999:
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. In addition, some test suites rely on specific users or groups. 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/log/{btmp,lastlog,faillog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664 /var/log/lastlog
chmod -v 600 /var/log/btmp
```

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

## Note

The `/run/utmp` file records the users that are currently logged in. This file is created dynamically in the boot scripts.

## 6.7. Linux-4.20.12 API 头文件

Linux API 头文件（在 linux-4.20.12.tar.xz 里）会将内核 API 导出给 Glibc 使用。

大致构建用时：    少于 0.1 SBU  
所需磁盘空间：    941 MB

### 6.7.1. Linux API 头文件的安装

Linux 内核需要提供一个应用编程接口（API）供系统的 C 库（LFS 中的 Glibc）调用。这通过整理 Linux 内核源码包中的多个 C 头文件来完成。

确保在之前的动作里没有留下旧文件和依赖关系：

```
make mrproper
```

现在要从源代码里解压出用户需要的内核头文件。因为解压过程会删除目标目录下所有文件，所以我们会先输出到一个本地中间目录后再拷贝到需要的地方。而且里面还有一些隐藏文件是给内核开发人员用的，而 LFS 不需要，所以会将它们从中间目录里删除。

```
make INSTALL_HDR_PATH=dest headers_install
find dest/include \( -name .install -o -name ..install.cmd \) -delete
cp -rv dest/include/* /usr/include
```

### 6.7.2. Linux API 头文件内容

安装的头文件：    /usr/include/asm/\*.h, /usr/include/asm-generic/\*.h, /usr/include/drm/\*.h, /usr/include/linux/\*.h, /usr/include/misc/\*.h, /usr/include/mtd/\*.h, /usr/include/rdma/\*.h, /usr/include/scsi/\*.h, /usr/include/sound/\*.h, /usr/include/video/\*.h, 和 /usr/include/xen/\*.h  
安装的目录：    /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, 和 /usr/include/xen

#### 简要介绍

/usr/include/asm/*.h	Linux API ASM 头文件
/usr/include/asm-generic/*.h	Linux API ASM 通用头文件
/usr/include/drm/*.h	Linux API DRM 头文件
/usr/include/linux/*.h	Linux API Linux 头文件
/usr/include/mtd/*.h	Linux API MTD 头文件
/usr/include/rdma/*.h	Linux API RDMA 头文件
/usr/include/scsi/*.h	Linux API SCSI 头文件
/usr/include/sound/*.h	Linux API 音频头文件
/usr/include/video/*.h	Linux API 视频头文件
/usr/include/xen/*.h	Linux API Xen 头文件

## 6.8. Man-pages-4.16

Man-pages 软件包里包含了超过 2,200 份 man 手册页面。

大致构建用时: 少于 0.1 SBU

所需磁盘空间: 28 MB

### 6.8.1. 安装 Man-pages

运行下面的命令安装 Man-pages :

```
make install
```

### 6.8.2. Man-pages 内容

安装的文件: 各种 man 手册页面

#### 简要介绍

`man pages` 描述 C 编程语言函数，重要的设备文件，以及主要的配置文件

## 6.9. Glibc-2.29

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.

大致构建用时: 22 SBU  
所需磁盘空间: 3.2 GB

### 6.9.1. Installation of Glibc

#### Note

The Glibc build system is self-contained and will install perfectly, even though the compiler specs file and linker are still pointing to `/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.

Some of the Glibc programs use the non-FHS compilant `/var/db` directory to store their runtime data. Apply the following patch to make such programs store their runtime data in the FHS-compliant locations:

```
patch -Np1 -i ../glibc-2.29-fhs-1.patch
```

First create a compatibility symlink to avoid references to `/tools` in our final glibc:

```
ln -sfv /tools/lib/gcc /usr/lib
```

Determine the GCC include directory and create a symlink for LSB compliance. Additionally, for `x86_64`, create a compatibility symlink required for the dynamic loader to function correctly:

```
case $(uname -m) in
  i?86)  GCC_INCDIR=/usr/lib/gcc/$(uname -m)-pc-linux-gnu/8.2.0/include
         ln -sfv ld-linux.so.2 /lib/ld-lsb.so.3
        ;;
  x86_64) GCC_INCDIR=/usr/lib/gcc/x86_64-pc-linux-gnu/8.2.0/include
         ln -sfv ../lib/ld-linux-x86-64.so.2 /lib64
         ln -sfv ../lib/ld-linux-x86-64.so.2 /lib64/ld-lsb-x86-64.so.3
        ;;
esac
```

Remove a file that may be left over from a previous build attempt:

```
rm -f /usr/include/limits.h
```

The Glibc documentation recommends building Glibc in a dedicated build directory:

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

Prepare Glibc for compilation:

```
CC="gcc -isystem $GCC_INCDIR -isystem /usr/include" \
../configure --prefix=/usr \
              --disable-werror \
              --enable-kernel=3.2 \
              --enable-stack-protector=strong \
              libc_cv_slibdir=lib
unset GCC_INCDIR
```

The meaning of the options and new configure parameters:

```
CC="gcc -isystem $GCC_INCDIR -isystem /usr/include"
```

Setting the location of both gcc and system include directories avoids introduction of invalid paths in debugging symbols.

```
--disable-werror
```

This option disables the `-Werror` option passed to GCC. This is necessary for running the test suite.

```
--enable-stack-protector=strong
```

This option increases system security by adding extra code to check for buffer overflows, such as stack smashing attacks.

```
libc_cv_slibdir=/lib
```

This variable sets the correct library for all systems. We do not want lib64 to be used.

Compile the package:

```
make
```

## Important

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

Generally a few tests do not pass. The test failures listed below are usually safe to ignore.

```
case $(uname -m) in
  i?86)  ln -sfv $PWD/elf/ld-linux.so.2      /lib ;;
  x86_64) ln -sfv $PWD/elf/ld-linux-x86-64.so.2 /lib ;;
esac
```

## Note

The symbolic link above is needed to run the tests at this stage of building in the chroot environment. It will be overwritten in the install phase below.

```
make check
```

You may see some test failures. The Glibc test suite is somewhat dependent on the host system. This is a list of the most common issues seen for some versions of LFS:

- misc/tst-ttname is known to fail in the LFS chroot environment.
- inet/tst-idna\_name\_classify is known to fail in the LFS chroot environment.
- posix/tst-getaddrinfo4 and posix/tst-getaddrinfo5 may fail on some architectures.
- The nss/tst-nss-files-hosts-multi test may fail for reasons that have not been determined.
- The rt/tst-cputimer{1,2,3} tests depend on the host system kernel. Kernels 4.14.91–4.14.96, 4.19.13–4.19.18, and 4.20.0–4.20.5 are known to cause these tests to fail.
- The math tests sometimes fail when running on systems where the CPU is not a relatively new Intel or AMD processor.

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
```

Fix the generated Makefile to skip an unneeded sanity check that fails in the LFS partial environment:

```
sed '/test-installation/s@$(PERL)@echo not running@' -i ../Makefile
```

Install the package:

```
make install
```

Install the configuration file and runtime directory for nscd:

```
cp -v ../nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

Install the systemd support files for nscd:

```
install -v -Dm644 ../nscd/nscd.tmpfiles /usr/lib/tmpfiles.d/nscd.conf
install -v -Dm644 ../nscd/nscd.service /lib/systemd/system/nscd.service
```

Next, install the locales that can make the system respond in a different language. None of the locales are required, but if some of them are missing, the test suites of 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/cs_CZ` charset-independent locale definition with the `/usr/share/i18n/charmaps/UTF-8.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 POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
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 el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f UTF-8 en_GB.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 it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SIJS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
```

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

Alternatively, install all locales listed in the `glibc-2.29/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-2.29/localedata/SUPPORTED` file in the unlikely case you need them.

## Note

Glibc now uses `libidn2` when resolving internationalized domain names. This is a run time dependency. If this capability is needed, the instructions for installing `libidn2` are in the BLFS `libidn2` page.

## 6.9.2. Configuring Glibc

### 6.9.2.1. Adding `nsswitch.conf`

The `/etc/nsswitch.conf` file needs to be created because the Glibc defaults do not work well in a networked environment.

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
```

### 6.9.2.2. Adding time zone data

Install and set up the time zone data with the following:

```
tar -xvf ../tzdata2018i.tar.gz

ZONEINFO=/usr/share/zoneinfo
mkdir -pv $ZONEINFO/{posix,right}

for tz in etcetera southamerica northamerica europe africa antarctica \
    asia australasia backward pacificnew systemv; do
    zic -L /dev/null -d $ZONEINFO ${tz}
    zic -L /dev/null -d $ZONEINFO/posix ${tz}
    zic -L leapseconds -d $ZONEINFO/right ${tz}
done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

The meaning of the `zic` commands:

`zic -L /dev/null ...`

This creates posix time zones, without any leap seconds. It is conventional to put these in both `zoneinfo` and `zoneinfo/posix`. It is necessary to put the POSIX time zones in `zoneinfo`, otherwise various test-suites will report errors. On an embedded system, where space is tight and you do not intend to ever update the time zones, you could save 1.9MB by not using the `posix` directory, but some applications or test-suites might produce some failures.

`zic -L leapseconds ...`

This creates right time zones, including leap seconds. On an embedded system, where space is tight and you do not intend to ever update the time zones, or care about the correct time, you could save 1.9MB by omitting the `right` directory.

`zic ... -p ...`

This creates the `posixrules` file. We use New York because POSIX requires the daylight savings time rules to be in accordance with US rules.

One way to determine the local time zone is to 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 time zones 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:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

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



### 6.9.2.3. 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

EOF
```

If desired, the dynamic loader can also search a directory and include the contents of files found there. Generally the files in this include directory are one line specifying the desired library path. To add this capability run the following commands:

```
cat >> /etc/ld.so.conf << "EOF"
# Add an include directory
include /etc/ld.so.conf.d/*.conf

EOF
mkdir -pv /etc/ld.so.conf.d
```

### 6.9.3. Contents of Glibc

Installed programs:	catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, and zic
Installed libraries:	ld-2.29.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, 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, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so, and libutil.{a,so}
Installed directories:	/usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/rpcsvc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/nscd, and /var/lib/nss_db

### 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

<code>makedb</code>	Creates a simple database from textual input
<code>mtrace</code>	Reads and interprets a memory trace file and displays a summary in human-readable format
<code>nscd</code>	A daemon that provides a cache for the most common name service requests
<code>pldd</code>	Lists dynamic shared objects used by running processes
<code>sln</code>	A statically linked ln program
<code>sotrust</code>	Traces shared library procedure calls of a specified command
<code>sprof</code>	Reads and displays shared object profiling data
<code>tzselect</code>	Asks the user about the location of the system and reports the corresponding time zone description
<code>xtrace</code>	Traces the execution of a program by printing the currently executed function
<code>zdump</code>	The time zone dumper
<code>zic</code>	The time zone compiler
<code>ld-2.29.so</code>	The helper program for shared library executables
<code>libBrokenLocale</code>	Used internally by Glibc as a gross hack to get broken programs (e.g., some Motif applications) running. See comments in <code>glibc-2.29/locale/broken_cur_max.c</code> for more information
<code>libSegFault</code>	The segmentation fault signal handler, used by <code>catchsegv</code>
<code>libanl</code>	An asynchronous name lookup library
<code>libc</code>	The main C library
<code>libcidn</code>	Used internally by Glibc for handling internationalized domain names in the <code>getaddrinfo()</code> function
<code>libcrypt</code>	The cryptography library
<code>libdl</code>	The dynamic linking interface library
<code>libg</code>	Dummy library containing no functions. Previously was a runtime library for g++
<code>libieee</code>	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
<code>libm</code>	The mathematical library
<code>libmcheck</code>	Turns on memory allocation checking when linked to
<code>libmemusage</code>	Used by <code>memusage</code> to help collect information about the memory usage of a program
<code>libnsl</code>	The network services library
<code>libnss</code>	The Name Service Switch libraries, containing functions for resolving host names, user names, group names, aliases, services, protocols, etc.
<code>libpthread</code>	The POSIX threads library
<code>libresolv</code>	Contains functions for creating, sending, and interpreting packets to the Internet domain name servers
<code>librpcsvc</code>	Contains functions providing miscellaneous RPC services
<code>librt</code>	Contains functions providing most of the interfaces specified by the POSIX.1b Realtime Extension
<code>libthread_db</code>	Contains functions useful for building debuggers for multi-threaded programs
<code>libutil</code>	Contains code for “standard” functions used in many different Unix utilities

## 6.10. 调整工具链

现在最后的 C 语言库已经装好了，是时候调整工具链，让新编译的程序链接到这些新的库上。

首先，备份 `/tools` 链接器，然后用我们在第五章调整过的链接器代替它。我们还会创建一个链接，链接到 `/tools/$(uname -m)-pc-linux-gnu/bin` 的副本：

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(uname -m)-pc-linux-gnu/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(uname -m)-pc-linux-gnu/bin/ld
```

接下来，修改 GCC 参数文件，让它指向新的动态连接器。只需删除所有「`/tools`」的实例，这样应该可以留下到达动态链接器的正确路径。还要调整参数文件，这样 GCC 就知道怎样找到正确的头文件和 Glibc 启动文件。一个 `sed` 命令就能完成这些：

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

直观地检查参数文件来确认预期的变化已确实完成是个好办法。

确保已调整的工具链的基本功能（编译和链接）都能如期进行是非常必要的。怎样做呢？执行下面这条命令：

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

如果没有任何错误，上条命令的输出应该是（不同的平台上的动态链接器可能名字不同）：

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

注意在 64 位系统中，虽然我们的动态链接位于 `/lib` 中，但却可以通过 `/lib64` 中的符号链接访问。

### Note

32 位系统的解释器应该是 `/lib/ld-linux.so.2`。

现在确保我们已经设置好了启动文件：

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

上一条命令的输出应该是：

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

确保链接器能找到正确的头文件：

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

这条命令应该返回如下输出：

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

接下来，确认新的链接器已经在使用正确的搜索路径：

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

应该忽略指向带有 `-linux-gnu` 的路径，上条命令的输出应该是：

```
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib")
```

然后我们要确定我们使用的是正确的 `libc`：

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

上条命令的输出应该为：

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

最后，确保 GCC 使用的是正确的动态链接器：

```
grep found dummy.log
```

上条命令的结果应该是（不同的平台上链接器名字可以不同）：

```
found ld-linux-x86-64.so.2 at /lib/ld-linux-x86-64.so.2
```

如果显示的结果不一样或者根本没有显示，那就出了大问题。检查并回溯之前的步骤，找到出错的地方并改正。最有可能的原因是参数文件的调整出了问题。在进行下一步之前所有的问题都要解决。

确保一切正常之后，清除测试文件：

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

## 6.11. Zlib-1.2.11

Zlib 软件包包括一些程序所使用的压缩和解压缩例程。

大致构建用时:               少于 0.1 SBU

所需磁盘空间:               4.4 MB

### 6.11.1. 安装 Zlib

准备编译 Zlib :

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

编译软件包 :

```
make
```

输入命令查看结果 :

```
make check
```

安装软件包:

```
make install
```

共享库需要移动到 `/lib` , 因此需要重建 `.so` 里面的 `/usr/lib` 文件 :

```
mv -v /usr/lib/libz.so.* /lib  
ln -sfv ../../lib/$(readlink /usr/lib/libz.so) /usr/lib/libz.so
```

### 6.11.2. Zlib 软件包内容

安装的库:                   libz.{a,so}

#### 简要介绍

libz 包括一些程序所使用的压缩和解压缩功能。

## 6.12. File-5.36

File 软件包包括一个判断给定的某个或某些文件文件类型的工具。

大致构建用时: 0.1 SBU

所需磁盘空间: 18 MB

### 6.12.1. 安装 File

准备编译 File :

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

编译软件包 :

```
make
```

输入命令检查结果 :

```
make check
```

安装软件包 :

```
make install
```

### 6.12.2. File 软件包内容

安装的程序: file

安装的库: libmagic.so

#### 简要介绍

file 为每个文件归类；这可以通过一些测试达到——文件系统测试，魔术数字测试还有语言测试。

libmagic 包含程序 file 进行魔术数字识别的例程。

## 6.13. Readline-8.0

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

大致构建用时: 0.1 SBU  
所需磁盘空间: 15 MB

### 6.13.1. 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
```

Prepare Readline for compilation:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/readline-8.0
```

Compile the package:

```
make SHLIB_LIBS="-L/tools/lib -lncursesw"
```

The meaning of the make option:

```
SHLIB_LIBS="-L/tools/lib -lncursesw"
```

This option forces Readline to link against the libncursesw library.

This package does not come with a test suite.

Install the package:

```
make SHLIB_LIBS="-L/tools/lib -lncursesw" install
```

Now move the dynamic libraries to a more appropriate location and fix up some permissions and symbolic links:

```
mv -v /usr/lib/lib{readline,history}.so.* /lib
chmod -v u+w /lib/lib{readline,history}.so.*
ln -sfv ../../lib/$(readlink /usr/lib/libreadline.so) /usr/lib/libreadline.so
ln -sfv ../../lib/$(readlink /usr/lib/libhistory.so) /usr/lib/libhistory.so
```

If desired, install the documentation:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.0
```

### 6.13.2. Contents of Readline

Installed libraries: libhistory.so and libreadline.so  
Installed directories: /usr/include/readline, /usr/share/readline, and /usr/share/doc/readline-8.0

#### Short Descriptions

**libhistory** Provides a consistent user interface for recalling lines of history

**libreadline** Provides a set of commands for manipulating text entered in an interactive session of a program.

## 6.14. M4-1.4.18

M4 软件包包含一个宏处理器。

大致构建用时: 0.4 SBU  
所需磁盘空间: 33 MB

### 6.14.1. 安装 M4

首先，对应 glibc-2.28 的需求做一些修复：

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' lib/*.c  
echo "#define _IO_IN_BACKUP 0x100" >> lib/stdio-impl.h
```

准备编译 M4：

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

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

用以下命令测试结果：

```
make install
```

### 6.14.2. M4 软件包内容

安装的程序: m4

#### 简要介绍

m4 复制给定的文件并扩展其中包括的宏[这些宏或者是内建的或是用户定义的，可以有任何数目的参数。除了进行宏扩展，m4 还有用于包含命名文件、运行 Unix 命令、进行整数运算、操作文本、递归等内建函数。m4 程序可以作为一个编译器的前端，也可以作为一个宏处理器使用。]



## 6.15. Bc-1.07.1

The Bc package contains an arbitrary precision numeric processing language.

大致构建用时: 0.1 SBU  
所需磁盘空间: 4.1 MB

### 6.15.1. Installation of Bc

First, change an internal script to use sed instead of ed:

```
cat > bc/fix-libmath_h << "EOF"
#!/bin/bash
sed -e '1 s/^{"/' \
    -e 's/$"/,' \
    -e '2,$ s/^{"/' \
    -e '$ d' \
    -i libmath.h

sed -e '$ s/$/0}/' \
    -i libmath.h
EOF
```

Create temporary symbolic links so the package can find the readline library and confirm that its required libncurses library is available. Even though the libraries are in /tools/lib at this point, the system will use /usr/lib at the end of this chapter.

```
ln -sv /tools/lib/libncursesw.so.6 /usr/lib/libncursesw.so.6
ln -sfv libncursesw.so.6 /usr/lib/libncurses.so
```

Fix an issue in configure due to missing files in the early stages of LFS:

```
sed -i -e '/flex/s/as_fn_error/: ;; # &/' configure
```

Prepare Bc for compilation:

```
./configure --prefix=/usr \
    --with-readline \
    --mandir=/usr/share/man \
    --infodir=/usr/share/info
```

The meaning of the configure options:

`--with-readline`

This option tells Bc to use the readline library that is already installed on the system rather than using its own readline version.

Compile the package:

```
make
```

To test bc, run the commands below. There is quite a bit of output, so you may want to redirect it to a file. There are a very small percentage of tests (10 of 12,144) that will indicate a round off error at the last digit.

```
echo "quit" | ./bc/bc -l Test/checklib.b
```

Install the package:

```
make install
```

### 6.15.2. Contents of Bc

Installed programs: bc and dc

#### Short Descriptions

bc A command line calculator

dc A reverse-polish command line calculator

## 6.16. Binutils-2.32

Binutils 软件包包含一个链接器、一个汇编器、以及其它处理目标文件的工具。

大致构建用时: 6.9 SBU  
所需磁盘空间: 4.9 GB

### 6.16.1. 安装 Binutils

通过一个简单测试验证在 chroot 环境下 PTY 工作正常：

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

这个命令应该输出以下内容：

```
spawn ls
```

假如输出包括下面的信息，那么表示没有为 PTY 操作设置好环境。在运行 Binutils 和 GCC 的测试套件之前需要解决这个问题：

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

Binutils 的文档建议在一个专用的编译目录中编译 Binutils：

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

Prepare Binutils for compilation:

```
../configure --prefix=/usr      \
              --enable-gold      \
              --enable-ld=default \
              --enable-plugins   \
              --enable-shared    \
              --disable-werror   \
              --enable-64-bit-bfd \
              --with-system-zlib
```

配置参数的意义：

- `--enable-gold`  
构建 gold 链接器并将其安装为 ld.gold（紧挨着默认链接器）。
- `--enable-ld=default`  
构建原来的 bfd 链接器并将其安装为 ld（默认链接器）和 ld.bfd。
- `--enable-plugins`  
为链接器启用插件支持。
- `--enable-64-bit-bfd`  
启用 64 位支持（针对字宽较窄的主机）。64 位系统可能没什么必要，但也不会有什么坏处。
- `--with-system-zlib`  
使用安装的 zlib 库替代自带的版本构建。

编译软件包：

```
make tooldir=/usr
```

make 参数的含义：

`tooldir=/usr`  
一般来说，tooldir（最终存放可执行文件的目录）设置为 `$(exec_prefix)/$(target_alias)`。例如，x86\_64 机器会把它扩展为 `/usr/x86_64-unknown-linux-gnu`。因为这是个自定制的系统，并不需要 `/usr` 中的特定目标目录。如果系统用于交叉编译（例如，在 Intel 机器上编译能生成在 PowerPC 机器上运行的代码的软件包）会使用 `$(exec_prefix)/$(target_alias)`。

### Important

本章节中的 Binutils 测试套件至关重要，任何情况下都不能跳过。

查看结果：

```
make -k check
```

已知一个测试 `debug_msg.sh` 会失败。

安装软件包：

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

## 6.16.2. Binutils 内容

安装的程序:	<code>addr2line</code> , <code>ar</code> , <code>as</code> , <code>c++filt</code> , <code>elfedit</code> , <code>gprof</code> , <code>ld</code> , <code>ld.bfd</code> , <code>ld.gold</code> , <code>nm</code> , <code>objcopy</code> , <code>objdump</code> , <code>ranlib</code> , <code>readelf</code> , <code>size</code> , <code>strings</code> , 和 <code>strip</code>
安装的库:	<code>libbfd.{a,so}</code> 和 <code>libopcodes.{a,so}</code>
安装目录:	<code>/usr/lib/ldscripts</code>

### 简要介绍

<code>addr2line</code>	转换程序地址为文件名称和行号；给定一个地址和可执行文件的名称，它使用可执行文件中的调试信息来判断与该地址关联的源文件以及行号。
<code>ar</code>	创建、更改以及抽取归档文件。
<code>as</code>	一个将 <code>gcc</code> 的输出汇编到目标文件的汇编器。
<code>c++filt</code>	链接器用来过滤 C++ 和 Java 符号以及防止重载函数冲突。
<code>elfedit</code>	更新 ELF 文件的 ELF 文件头。
<code>gprof</code>	显示调用关系图配置数据。
<code>ld</code>	一个将多个目标文件和归档文件合并为单一文件，重定位数据及绑定符号引用的链接器。
<code>ld.gold</code>	一个阉割版的 <code>ld</code> ，仅支持 <code>elf</code> 对象文件格式
<code>ld.bfd</code>	到 <code>ld</code> 的硬链接。
<code>nm</code>	列出指定目标文件中出现的符号。
<code>objcopy</code>	转换某种类型的目标文件到另一种类型。
<code>objdump</code>	显示给定目标文件的信息，用选项可以控制显示特定信息；显示的信息对于使用编译工具的程序员非常有用。
<code>ranlib</code>	生成归档文件内容的索引并保存到归档文件；索引列出了所有归档文件成员——可重定位的目标文件定义的符号。
<code>readelf</code>	显示 ELF 类型的二进制文件的信息。
<code>size</code>	列出所给目标文件各部分大小和总的大小。
<code>strings</code>	对每个给定文件，输出不低于指定长度（默认是4）的可打印字符序列；对于目标文件，它默认只打印初始化和引导部分的字符串，而对于其它类型的文件扫描整个文件。
<code>strip</code>	从目标文件中去除符号。
<code>libbfd</code>	二进制文件描述库。
<code>libopcodes</code>	一个库用于处理操作码——「可读文本」版的处理器指令；用于构建类似 <code>objdump</code> 的工具。

## 6.17. GMP-6.1.2

GMP 软件包包含一些数学库。这里有对任意精度数值计算很有用的函数。

大致构建用时: 1.3 SBU  
所需磁盘空间: 61 MB

### 6.17.1. 安装 GMP

#### Note

如果你是为 32 位的 x86 系统编译，但是你的 CPU 可以运行 64 位代码并且环境中你有指定的 CFLAGS，那么配置脚本会尝试配置为 64 位并导致失败。用下面的方式执行配置命令来避免这个问题：

```
ABI=32 ./configure ...
```

#### Note

GMP 的默认设定会为主机的处理器优化库。如果你不需要完美符合主机 CPU 的库，可以通过下方命令创建通用库，这样的话契合度会差一些：

```
cp -v configfsf.guess config.guess
cp -v configfsf.sub config.sub
```

准备编译 GMP：

```
./configure --prefix=/usr \
            --enable-cxx \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.1.2
```

新配置选项的含义：

```
--enable-cxx
    这个参数启用 C++ 支持

--docdir=/usr/share/doc/gmp-6.1.2
    这个变量指定保存文档的正确位置
```

编译软件包并生成 HTML 文档：

```
make
make html
```

#### Important

该章节 GMP 的测试套件至关重要，任何情况下都不能跳过。

查看结果：

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

#### Caution

GMP 中的代码对于其构建的处理器进行了高度优化。有时，检测处理器的代码会误认系统的功能，并在测试中报错，或是在其他应用使用 GMP 库的时候显示消息「Illegal instruction (非法指令)」。在这种情况下，GMP 需要重新配置选项 `--build=x86_64-unknown-linux-gnu` 并重新构建。

确认测试套件中所有的 190 个测试都通过了。通过输入下面的命令检查结果：

```
awk '/# PASS:/{total+= $3} ; END{print total}' gmp-check-log
```

安装软件包和文档：

```
make install  
make install-html
```

## 6.17.2. GMP 内容

安装的库: libgmp.so 和 libgmpxx.so  
安装目录: /usr/share/doc/gmp-6.1.2

### 简要介绍

libgmp 包括精度数学函数  
libgmpxx 包括 C++ 精度属性函数

## 6.18. MPFR-4.0.2

MPFR 软件包包含多精度数学函数。

大致构建用时: 1.0 SBU

所需磁盘空间: 37 MB

### 6.18.1. 安装 MPFR

准备编译 MPFR :

```
./configure --prefix=/usr \
            --disable-static \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.0.2
```

编译软件包并生成 HTML 文档 :

```
make
make html
```

#### Important

该章节 MPFR 的测试套件至关重要，任何情况下都不能跳过。

检查结果确认通过了所有的测试 :

```
make check
```

安装软件包以及文档 :

```
make install
make install-html
```

### 6.18.2. MPFR 软件包内容

安装的库: libmpfr.so

安装目录: /usr/share/doc/mpfr-4.0.2

#### 简要介绍

libmpfr 包含多精度数学函数

## 6.19. MPC-1.1.0

MPC 软件包包含一个能以任意高精度进行复数数值计算和对结果进行正确四舍五入的库。

大致构建用时: 0.3 SBU  
所需磁盘空间: 22 MB

### 6.19.1. 安装 MPC

准备编译 MPC :

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.1.0
```

编译软件包并生成 HTML 文档 :

```
make
make html
```

用以下命令检查结果 :

```
make check
```

安装软件包以及它的帮助文档 :

```
make install
make install-html
```

### 6.19.2. MPC 软件包内容

安装的库: libmpc.so  
安装目录: /usr/share/doc/mpc-1.1.0

#### 简要介绍

libmpc 包含复数数学函数

## 6.20. Shadow-4.6

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

大致构建用时: 0.2 SBU

所需磁盘空间: 46 MB

### 6.20.1. Installation of Shadow

#### Note

If you would like to enforce the use of strong passwords, refer to <http://www.linuxfromscratch.org/blfs/view/8.4/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. Also Prevent the installation of manual pages that were already installed in Section#6.8, “Man-pages-4.16”.

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.5 / /' {} \;
```

Instead of using the default crypt method, use the more secure SHA-512 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 SHA512@' \
-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
```

Make a minor change to make the first group number generated by useradd 1000:

```
sed -i 's/1000/999/' etc/useradd
```

Prepare Shadow for compilation:

```
./configure --sysconfdir=/etc --with-group-name-max-length=32
```

The meaning of the configure option:

`--with-group-name-max-length=32`

The maximum user name is 32 characters. Make the maximum group name the same.

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
```



## 6.20.2. 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 behavior is undesirable, 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
```

## 6.20.3. Setting the root password

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

```
passwd root
```

## 6.20.4. Contents of Shadow

Installed programs:	<code>chage</code> , <code>chfn</code> , <code>chgpaswd</code> , <code>chpasswd</code> , <code>chsh</code> , <code>expiry</code> , <code>faillog</code> , <code>gpaswd</code> , <code>groupadd</code> , <code>groupdel</code> , <code>groupmems</code> , <code>groupmod</code> , <code>grpck</code> , <code>grpconv</code> , <code>grpunconv</code> , <code>lastlog</code> , <code>login</code> , <code>logoutd</code> , <code>newgidmap</code> , <code>newgrp</code> , <code>newuidmap</code> , <code>newusers</code> , <code>nologin</code> , <code>passwd</code> , <code>pwck</code> , <code>pwconv</code> , <code>pwunconv</code> , <code>sg</code> (link to <code>newgrp</code> ), <code>su</code> , <code>useradd</code> , <code>userdel</code> , <code>usermod</code> , <code>vigr</code> (link to <code>vipw</code> ), and <code>vipw</code>
Installed directory:	<code>/etc/default</code>

### Short Descriptions

<code>chage</code>	Used to change the maximum number of days between obligatory password changes
<code>chfn</code>	Used to change a user's full name and other information
<code>chgpaswd</code>	Used to update group passwords in batch mode
<code>chpasswd</code>	Used to update user passwords in batch mode
<code>chsh</code>	Used to change a user's default login shell
<code>expiry</code>	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 <code>/etc/group</code> and <code>/etc/gshadow</code>
grpconv	Creates or updates the shadow group file from the normal group file
grpunconv	Updates <code>/etc/group</code> from <code>/etc/gshadow</code> and then deletes the latter
lastlog	Reports the most recent login of all users or of a given user
login	Is used by the system to let users sign on
logoutd	Is a daemon used to enforce restrictions on log-on time and ports
newgidmap	Is used to set the gid mapping of a user namespace
newgrp	Is used to change the current GID during a login session
newuidmap	Is used to set the uid mapping of a user namespace
newusers	Is used to create or update an entire series of user accounts
nologin	Displays a message that an account is not available; it is designed to be used as the default shell for accounts that have been disabled
passwd	Is used to change the password for a user or group account
pwck	Verifies the integrity of the password files <code>/etc/passwd</code> and <code>/etc/shadow</code>
pwconv	Creates or updates the shadow password file from the normal password file
pwunconv	Updates <code>/etc/passwd</code> from <code>/etc/shadow</code> and then deletes the latter
sg	Executes a given command while the user's GID is set to that of the given group
su	Runs a shell with substitute user and group IDs
useradd	Creates a new user with the given name, or updates the default new-user information
userdel	Deletes the given user account
usermod	Is used to modify the given user's login name, User Identification (UID), shell, initial group, home directory, etc.
vigr	Edits the <code>/etc/group</code> or <code>/etc/gshadow</code> files
vipw	Edits the <code>/etc/passwd</code> or <code>/etc/shadow</code> files

## 6.21. GCC-8.2.0

GCC 软件包包括 GNU 编译器集，其中有 C 和 C++ 的编译器。

大致构建用时: 92 SBU (with tests)

所需磁盘空间: 3.9 GB

### 6.21.1. 安装 GCC

如果是在 x86\_64 上实施构建，更改 64 位库的默认目录名为「lib」：

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
    ;;
esac
```

删除之前创建的符号链接，因为最终 gcc 将被安装在这：

```
rm -f /usr/lib/gcc
```

GCC 的文档建议在源代码目录之外一个专用的编译目录中编译 GCC：

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

准备编译 GCC：

```
SED=sed \
../configure --prefix=/usr \
              --enable-languages=c,c++ \
              --disable-multilib \
              --disable-bootstrap \
              --disable-libmpx \
              --with-system-zlib
```

注意，对于其它的编程语言，现在还有一些前提条件没有准备好。可以查看 BLFS Book 了解如何编译 GCC 支持的所有语言的指令。

新配置选项的含义：

SED=sed

设置环境变量防止访问到硬编码的 /tools/bin/sed 路径。

--disable-libmpx

这个选项告知 GCC 不要编译 mpx (Memory Protection Extensions：内存保护扩展)。此特性在一些处理器上存在问题，已经在下一个版本的 GCC 中移除。

--with-system-zlib

这个选项告诉 GCC 链接系统安装的 Zlib 库，而不是它内部自带的库。

编译软件包：

```
make
```

### Important

本章节中 GCC 的测试套件至关重要，任何情况下都不能跳过。

GCC 测试套件中一个测试集的会耗尽堆栈空间，因此运行测试之前要增加栈大小：

```
ulimit -s 32768
```

删除已知会导致问题的测试：

```
rm ../gcc/testsuite/g++.dg/pr83239.C
```

以非特权用户测试编译结果，不要因为出现错误就停下来：

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

要查看测试套件结果的概要，运行：

```
../contrib/test_summary
```

如果仅查看摘要，则使用管道 `grep -A7 Summ` 选项控制输出将输出：

```
../contrib/test_summary | grep -A7 Summ
```

结果可以和 <http://www.linuxfromscratch.org/lfs/build-logs/8.4/> 以及 <http://gcc.gnu.org/ml/gcc-testresults/> 上的相比较。

一些意料之外的错误总是难以避免。GCC 开发者通常会意识到这些问题，但还没有解决。除非测试结果和上面 URL 中的相差很大，不然就可以安全继续。

## Note

在某些内核配置和 AMD 处理器组合的时候，肯能会在 `gcc.target/i386/mpx` 测试（旨在测试最新英特尔处理器上的 MPX 选项）中出现 1100 个失败。AMD 处理器的话可以安心的忽略这些。

安装软件包：

```
make install
```

创建 FHS 因为「历史」原因而需要的软链接。

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

很多软件包用命令 `cc` 调用 C 编译器。为了满足这些软件包，创建一个符号链接：

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

增加一个兼容符号链接启用编译程序时进行链接时间优化（Link Time Optimization，LTO）：

```
install -v -dm755 /usr/lib/bfd-plugins
ln -sfv ../../libexec/gcc/${gcc -dumpmachine}/8.2.0/liblto_plugin.so \
    /usr/lib/bfd-plugins/
```

现在我们最终的工具链已经准备就绪了，再一次确认编译和链接都能像预期那样正常工作很重要。我们通过做和前面章节做过的相同的完整性检查做到这点：

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

如果没有任何错误，上条命令的输出应该是（不同的平台上的动态链接器可能名字不同）：

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

现在确保我们已经设置好了启动文件：

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

上一条命令的输出应该是：

```
/usr/lib/gcc/x86_64-pc-linux-gnu/8.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/8.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/8.2.0/../../../../lib/crtn.o succeeded
```

取决于你机器的架构，上面的结果可能有稍微不同，差异通常是 `/usr/lib/gcc` 后目录的名称。这里重要的一点是 `gcc` 能在 `/usr/lib` 目录下找到所有的三个 `crt*.o` 文件。

确保链接器能找到正确的头文件：

```
grep -B4 '^ /usr/include' dummy.log
```

这条命令应该返回如下输出：

```
#include <...> search starts here:
/usr/lib/gcc/x86_64-pc-linux-gnu/8.2.0/include
/usr/local/include
/usr/lib/gcc/x86_64-pc-linux-gnu/8.2.0/include-fixed
/usr/include
```

同时，注意你的目标系统三段式后面的目录名称可能和上面的不同，这取决于你的架构。

接下来，确认新的链接器已经在使用正确的搜索路径：

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

应该忽略指向带有 '-linux-gnu' 的路径，上条命令的输出应该是：

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

32 位的系统可能有一些不同的目录。例如，下面是一台 i686 机器的输出：

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib32")
SEARCH_DIR("/usr/local/lib32")
SEARCH_DIR("/lib32")
SEARCH_DIR("/usr/lib32")
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

然后我们要确定我们使用的是正确的 libc：

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

上条命令的输出应该为：

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

最后，确保 GCC 使用的是正确的动态链接器：

```
grep found dummy.log
```

上条命令的结果应该是（不同的平台上链接器名字可以不同）：

```
found ld-linux-x86-64.so.2 at /lib/ld-linux-x86-64.so.2
```

如果显示的结果不一样或者根本没有显示，那就出了大问题。检查并回溯之前的步骤，找到出错的地方并改正。最有可能的原因是参数文件的调整出了问题。在进行下一步之前所有的问题都要解决。

确保一切正常之后，清除测试文件：

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

最后，移动位置放错的文件：

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

## 6.21.2. GCC 软件包内容

安装的程序：	c++, cc (链接到 gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, 和 gcov
安装的库：	libasan.{a,so}, libatomic.{a,so}, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libiberty.a, libitm.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.a, libsupc++.a, 和 libtsan.{a,so}
安装目录：	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, 和 /usr/share/gcc-8.2.0

### 简要介绍

c++	C++ 编译器
cc	C 编译器
cpp	C 预处理器；编译器用来扩展源文件中 #include、#define 以及类似语句
g++	C++ 编译器
gcc	C 编译器

gcc-ar	增加插件到命令行的 ar 的封装。这个程序只用于添加 "链接时间优化", 在使用默认编译选项时不起作用
gcc-nm	增加插件到命令行的 nm 的封装。这个程序只用于添加 "链接时间优化", 在使用默认编译选项时不起作用
gcc-ranlib	增加插件到命令行的 ranlib 的封装。这个程序只用于添加 "链接时间优化", 在使用默认编译选项时不起作用
gcov	一个覆盖测试工具; 用于分析程序以决定在哪里进行优化有最大的效果
libasan	Address Sanitizer (地址消毒剂) 运行时库。
libgcc	包含用于 gcc 的运行时支持
libgcov	当指示 GCC 启用分析时该库会被链接到程序中
libgomp	用于 C/C++、Fortran 语言的多平台共享内存并行编程的 OpenMP API 的 GNU 实现
libiberty	包含多种 GNU 程序所使用的例程, 包括 getopt, obstack, strerror, strtol, 和 strtoul
liblto_plugin	GCC 的链接时间优化 (LTO) 插件, 允许 GCC 跨编译单元进行优化
libquadmath	GCC 四精度数学库 API
libssp	包含支持 GCC 堆栈溢出保护功能的例程
libstdc++	标准 C++ 库
libsupc++	为 C++ 编程语言提供支持例程
libtsan	Thread Sanitizer (数据速率检测工具) 运行时库

## 6.22. Bzip2-1.0.6

Bzip2 软件包包含压缩和解压缩的程序。用 bzip2 压缩文本文件能获得比传统的 gzip 更好的压缩比。

大致构建用时：    少于 0.1 SBU  
所需磁盘空间：    2.3 MB

### 6.22.1. 安装 Bzip2

使用能为这个软件包安装帮助文档的补丁：

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch
```

下面的命令确保安装的符号链接是相对链接：

```
sed -i 's@(\ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

确认 man 页面安装到了正确的位置：

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

准备编译 Bzip2：

```
make -f Makefile-libbz2_so
make clean
```

make 参数的含义：

`-f Makefile-libbz2_so`

这会使用不同的 Makefile 文件编译 Bzip2，在这里是 Makefile-libbz2\_so，它会创建动态 libbz2.so 库，并把它链接到 Bzip2 工具。

编译并测试软件包：

```
make
```

安装程序：

```
make PREFIX=/usr install
```

安装使用动态链接库的 bzip2 二进制文件到 /bin 目录，创建一些必须的符号链接并清理：

```
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
```

### 6.22.2. Bzip2 软件包内容

安装的程序：    bunzip2 (链接到 bzip2), bzcat (链接到 bzip2), bzcmp (链接到 bzdiff), bzdiff, bzegrep (链接到 bzgrep), bzfgrep (链接到 bzgrep), bzgrep, bzip2, bzip2recover, bzless (链接到 bzmored), 和 bzmored

安装的库：    libbz2.{a,so}

安装目录：    /usr/share/doc/bzip2-1.0.6

#### 简要介绍

bunzip2	解压 bzip 压缩的文件
bzcat	解压到标准输出
bzcmp	对 bzip 压缩的文件运行 cmp 命令
bzdiff	对 bzip 压缩的文件运行 diff 命令
bzegrep	对 bzip 压缩的文件运行 egrep 命令
bzfgrep	对 bzip 压缩的文件运行 fgrep 命令
bzgrep	对 bzip 压缩的文件运行 grep 命令
bzip2	使用哈夫曼编码的 Burrows-Wheeler 块排序文本压缩算法压缩文件；压缩率比传统的用「Lempel-Ziv」算法的压缩器要好，比如 gzip。

<code>bzip2recover</code>	尝试从损坏的 bzip 压缩文件中恢复数据
<code>bzless</code>	对 bzip 压缩的文件运行 <code>less</code> 命令
<code>bzmore</code>	对 bzip 压缩的文件运行 <code>more</code> 命令
<code>libbz2</code>	用 Burrows-Wheeler 算法实现的无损的块排序数据压缩库



## 6.23. Pkg-config-0.29.2

pkg-config 软件包包含一个在配置和 make 文件运行时把 include 路径和库路径传递给编译工具的工具。

大致构建用时: 0.3 SBU  
所需磁盘空间: 30 MB

### 6.23.1. 安装 Pkg-config

准备编译 Pkg-config :

```
./configure --prefix=/usr \
            --with-internal-glib \
            --disable-host-tool \
            --docdir=/usr/share/doc/pkg-config-0.29.2
```

新配置选项的含义：

*--with-internal-glib*

这会让 pkg-config 使用它自己内部版本的 Glib，因为在 LFS 中没有可用的外部版本。

*--disable-host-tool*

此选项取消创建到 pkg-config 程序的不必要的硬链接。

编译软件包：

```
make
```

用以下命令检查结果：

```
make check
```

安装软件包：

```
make install
```

### 6.23.2. pkg-config 软件包内容

安装的软件: pkg-config  
安装目录: /usr/share/doc/pkg-config-0.29.2

#### 简要介绍

pkg-config 返回指定库或软件包的元信息

## 6.24. Ncurses-6.1

Ncurses 软件包包含用于不依赖于特定终端的字符屏幕处理的库。

大致构建用时: 0.3 SBU  
所需磁盘空间: 42 MB

### 6.24.1. 安装 Ncurses

不要安装静态库，它不受配置控制：

```
sed -i '/LIBTOOL_INSTALL/d' c++/Makefile.in
```

准备编译 Ncurses：

```
./configure --prefix=/usr \
            --mandir=/usr/share/man \
            --with-shared \
            --without-debug \
            --without-normal \
            --enable-pc-files \
            --enable-widec
```

新配置选项的含义：

*--enable-widec*

这个选项会编译宽字符库（例如 `libncursesw.so.6.1`）来代替普通字符库（例如 `libncurses.so.6.1`）。宽字符库可用于多字节和传统的 8 位本地字符，而常规的库只能用于 8 位本地字符。宽字符库和常规的库是源文件兼容的，而不是二进制文件兼容的。

*--enable-pc-files*

该选项为 `pkg-config` 生成和安装 `.pc` 文件。

*--without-normal*

该选项用于禁用构建和安装大多数的静态库。

编译软件包：

```
make
```

该软件包有个测试套件，但只能在安装完软件包后运行。测试程序在 `test/` 目录中。查看该目录中的 `README` 文件获取更详细信息。

安装软件包：

```
make install
```

移动共享库到期望的 `/lib` 文件夹：

```
mv -v /usr/lib/libncursesw.so.6* /lib
```

由于库已经被移走了，符号链接指向了一个不存在的文件。重建符号链接：

```
ln -sfv ../../lib/${readlink /usr/lib/libncursesw.so} /usr/lib/libncursesw.so
```

很多应用程序仍然希望编辑器能找到非宽字符的 Ncurses 库。通过符号链接和链接器脚本欺骗这样的应用链接到宽字符库：

```
for lib in ncurses form panel menu ; do
    rm -vf /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc /usr/lib/pkgconfig/${lib}.pc
done
```

最后，确保在编译时会查找 `-lncurses` 的旧应用程序仍然可以编译：

```
rm -vf /usr/lib/libcursesw.so
echo "INPUT(-lncurses)" > /usr/lib/libcursesw.so
ln -sfv libcurses.so /usr/lib/libcurses.so
```

如果需要的话，安装 Ncurses 的帮助文档：

```
mkdir -v /usr/share/doc/ncurses-6.1
cp -v -R doc/* /usr/share/doc/ncurses-6.1
```

## Note

上面的指令并不会创建非宽字符 Ncurses 库，因为没有从源文件中编译安装的软件包会在运行时链接它们。然而，已知的仅有二进制应用程序并能链接到非等宽字符的库，需要第 5 版的支持。如果你由于一些仅有二进制的程序或要和 LSB 兼容而必须要有这样的库，用下面的命令重新编译软件包：

```
make distclean
./configure --prefix=/usr \
            --with-shared \
            --without-normal \
            --without-debug \
            --without-cxx-binding \
            --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

## 6.24.2. Ncurses 软件包内容

安装的程序:	captainfo (链接到 tic), clear, infocmp, infotocap (链接到 tic), ncursesw6-config, reset (链接到 tset), tabs, tic, toe, tput, 和 tset
安装的库:	libcursesw.so (到 libncursesw.so 的符号链接和链接器脚本), libformw.so, libmenuw.so, libncursesw.so, libncurses++w.a, libpanelw.so, 以及库名称中没有 "w" 的对应的非宽字符部分。
安装目录:	/usr/share/tabset, /usr/share/terminfo, 和 /usr/share/doc/ncurses-6.1

## 简要介绍

captainfo	转换 termcap 描述为 terminfo 描述
clear	如果可以的话清空屏幕
infocmp	比较或输出 terminfo 描述
infotocap	转换 terminfo 描述为 termcap 描述
ncursesw6-config	P为 ncurses 提供配置信息
reset	重新初始化终端为默认设置
tabs	清空终端并设置制表符长度
tic	将 terminfo 文件从源文件格式转换到二进制格式的 terminfo 条目描述编译器需要 ncurses 例程 [terminfo 文件包含特定终端的功能信息]
toe	列出所有可用的终端类型，给出每个主名称和描述
tput	可以在 shell 中使用终端特定的功能值；也可用来重置或初始化终端或者报告它的完整名称
tset	可以用来初始化终端
libcursesw	到 libncursesw 的链接
libncursesw	包含在一个终端屏幕以多种复杂方式显示文本的函数；使用这些功能的一个好的例子是内核 make menuconfig 时的菜单显示
libformw	包含实现表单的函数
libmenuw	包含实现菜单的函数
libpanelw	包含实现面板的函数

## 6.25. Attr-2.4.48

The attr package contains utilities to administer the extended attributes on filesystem objects.

大致构建用时:                少于 0.1 SBU  
所需磁盘空间:                4.2 MB

### 6.25.1. Installation of Attr

Prepare Attr for compilation:

```
./configure --prefix=/usr \
            --disable-static \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.4.48
```

Compile the package:

```
make
```

The tests need to be run on a filesystem that supports extended attributes such as the ext2, ext3, or ext4 filesystems. To test the results, issue:

```
make check
```

Install the package:

```
make install
```

The shared library needs to be moved to `/lib`, and as a result the `.so` file in `/usr/lib` will need to be recreated:

```
mv -v /usr/lib/libattr.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libattr.so) /usr/lib/libattr.so
```

### 6.25.2. Contents of Attr

Installed programs:        attr, getfattr, and setfattr  
Installed library:        libattr.so  
Installed directories:    /usr/include/attr and /usr/share/doc/attr-2.4.48

### Short Descriptions

attr	Extends attributes on filesystem objects
getfattr	Gets the extended attributes of filesystem objects
setfattr	Sets the extended attributes of filesystem objects
libattr	Contains the library functions for manipulating extended attributes

## 6.26. Acl-2.2.53

Acl 软件包包含管理访问控制列表的工具，访问控制列表用于定义文件和目录更细粒度的自定义访问权限。

大致构建用时:               少于 0.1 SBU  
所需磁盘空间:               6.4 MB

### 6.26.1. 安装 Acl

准备编译 Acl :

```
./configure --prefix=/usr      \
            --disable-static   \
            --libexecdir=/usr/lib \
            --docdir=/usr/share/doc/acl-2.2.53
```

编译软件包 :

```
make
```

在用 Acl 库构建 Coreutils 后，Acl 测试才能在支持访问控制的文件系统上运行。如果需要的话，可以在本章后面构建完 Coreutils 之后回到这个软件包运行 `make check` 进行测试。

安装软件包 :

```
make install
```

需要移动共享库到 `/lib`，因此需要重建 `/usr/lib` 中的 `.so` 文件：

```
mv -v /usr/lib/libacl.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libacl.so) /usr/lib/libacl.so
```

### 6.26.2. Acl 软件包内容

安装的程序:               chacl, getfacl, 和 setfacl  
安装的库:                 libacl.so  
安装目录:                 /usr/include/acl 和 /usr/share/doc/acl-2.2.53

#### 简要介绍

chacl	更改文件或目录的访问控制列表
getfacl	获取文件访问控制列表
setfacl	设置文件访问控制列表
libacl	包括用于管理访问控制列表的库函数

## 6.27. Libcap-2.26

The Libcap package implements the user-space interfaces to the POSIX 1003.1e capabilities available in Linux kernels. These capabilities are a partitioning of the all powerful root privilege into a set of distinct privileges.

大致构建用时:           少于 0.1 SBU  
所需磁盘空间:           1.4 MB

### 6.27.1. Installation of Libcap

Prevent a static library from being installed:

```
sed -i '/install.*STALIBNAME/d' libcap/Makefile
```

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make RAISE_SETFCAP=no lib=lib prefix=/usr install  
chmod -v 755 /usr/lib/libcap.so.2.26
```

The meaning of the make option:

*RAISE\_SETFCAP=no*

This parameter skips trying to use setcap on itself. This avoids an installation error if the kernel or file system does not support extended capabilities.

*lib=lib*

This parameter installs the library in `$prefix/lib` rather than `$prefix/lib64` on `x86_64`. It has no effect on `x86`.

The shared library needs to be moved to `/lib`, and as a result the `.so` file in `/usr/lib` will need to be recreated:

```
mv -v /usr/lib/libcap.so.* /lib  
ln -sfv ../../lib/$(readlink /usr/lib/libcap.so) /usr/lib/libcap.so
```

### 6.27.2. Contents of Libcap

Installed programs:       capsh, getcap, getpcaps, and setcap  
Installed library:       libcap.so

#### Short Descriptions

capsh	A shell wrapper to explore and constrain capability support
getcap	Examines file capabilities
getpcaps	Displays the capabilities on the queried process(es)
setcap	Sets file capabilities
libcap	Contains the library functions for manipulating POSIX 1003.1e capabilities

## 6.28. Sed-4.7

Sed 软件包包含一个流编辑器。

大致构建用时: 0.3 SBU  
所需磁盘空间: 32 MB

### 6.28.1. 安装 Sed

首先修复 LFS 环境中的问题，然后移除一个失败的测试：

```
sed -i 's/usr/tools/' build-aux/help2man
sed -i 's/testsuite.panic-tests.sh/' Makefile.in
```

准备编译 Sed：

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

编译软件包并生成 HTML 文档：

```
make
make html
```

输入命令查看结果：

```
make check
```

安装软件包和它的文档：

```
make install
install -d -m755 /usr/share/doc/sed-4.7
install -m644 doc/sed.html /usr/share/doc/sed-4.7
```

### 6.28.2. Sed 软件包内容

安装的程序: sed  
安装目录: /usr/share/doc/sed-4.7

#### 简要介绍

sed 过滤器，一次性转换文本文件

## 6.29. Psmisc-23.2

Psmisc 软件包包含用于显示运行中进程信息的程序。

大致构建用时:               少于 0.1 SBU  
所需磁盘空间:               4.5 MB

### 6.29.1. 安装 Psmisc

准备编译 Psmisc :

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

编译软件包 :

```
make
```

该软件包没有测试套件。

安装软件包 :

```
make install
```

最后，将程序 killall 和 fuser 移动到 FHS 指定的位置 :

```
mv -v /usr/bin/fuser /bin
mv -v /usr/bin/killall /bin
```

### 6.29.2. Psmisc 软件包内容

安装的程序:               fuser, killall, peekfd, prtstat, pstree, 和 pstree.x11 ([链接到 pstree](#))

#### 简要介绍

fuser	报告使用指定文件或文件系统的进程的进程 ID (PID)
killall	根据名称杀死进程；它发送信号到所有的运行任何给定命令的进程
peekfd	根据 PID 查看正在运行进程的文件描述符
prtstat	打印关于某个进程的信息
pstree	以树形结构显示运行中的进程
pstree.x11	和 pstree 命令相同，但退出时它会等待确认



## 6.30. Iana-Etc-2.30

Iana-Etc 软件包为网络服务和协议提供数据。

大致构建用时:                少于 0.1 SBU

所需磁盘空间:                2.3 MB

### 6.30.1. 安装 Iana-Etc

下面的命令将 IANA 提供的原始数据转换为 `/etc/protocols` 和 `/etc/services` 数据文件的正确格式：

```
make
```

该软件包没有测试套件。

安装软件包：

```
make install
```

### 6.30.2. Iana-Etc 软件包内容

安装的文件:                `/etc/protocols` 和 `/etc/services`

#### 简要介绍

`/etc/protocols`    描述 TCP/IP 子系统中可用的多种 DARPA 网络协议

`/etc/services`    提供友好文本名称和背后分配的端口号以及协议类型之间的映射

## 6.31. Bison-3.3.2

The Bison package contains a parser generator.

大致构建用时: 0.3 SBU  
所需磁盘空间: 37 MB

### 6.31.1. Installation of Bison

Prepare Bison for compilation:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.3.2
```

Compile the package:

```
make
```

There is a circular dependency between bison and flex with regard to the checks. If desired, after installing flex in the next section, the bison package can be rebuilt and the bison checks can be run with `make check`.

Install the package:

```
make install
```

### 6.31.2. Contents of Bison

Installed programs: bison and yacc  
Installed library: liby.a  
Installed directory: /usr/share/bison

#### Short Descriptions

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

## 6.32. Flex-2.6.4

Flex 软件包包括一个用于生成识别文本模式的程序的工具。

大致构建用时: 0.4 SBU  
所需磁盘空间: 35 MB

### 6.32.1. 安装 Flex

首先，修复一个 glibc-2.26 引入的问题：

```
sed -i "/math.h/a #include <malloc.h>" src/flexdef.h
```

构建过程假设能使用程序 help2man 的 --help 选项来创建 man 手册。但这显然是不存在的，所以我们使用环境变量来跳过这步。现在，准备编译 Flex：

```
HELP2MAN=/tools/bin/true \  
./configure --prefix=/usr --docdir=/usr/share/doc/flex-2.6.4
```

编译软件包：

```
make
```

用以下命令测试结果（大约 0.5 SBU）：

```
make check
```

安装软件包：

```
make install
```

一些程序还不知道 flex 并尝试运行它的预处理器 lex。为了支持这些程序，创建以 lex 仿真模式运行 flex 的符号链接 lex：

```
ln -sv flex /usr/bin/lex
```

### 6.32.2. Flex 软件包内容

安装的程序: flex, flex++ (链接到 flex), 和 lex (链接到 flex)  
安装的库: libfl.so  
安装目录: /usr/share/doc/flex-2.6.4

#### 简要介绍

flex	一个用于生成能识别文本模式程序的工具；它允许指定多种用于模式发现的规则，从而消除了开发专门程序的需要
flex++	flex 的扩展，用于生成 C++ 代码和类。是到 flex 的符号链接
lex	一个以 flex 仿真模式运行 lex 的符号链接。
libfl	flex 库

## 6.33. Grep-3.3

Grep 软件包包含用于在文件中搜索的程序。

大致构建用时: 0.4 SBU  
所需磁盘空间: 37 MB

### 6.33.1. 安装 Grep

准备编译 Grep :

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

编译软件包 :

```
make
```

用以下命令测试结果 :

```
make -k check
```

安装软件包 :

```
make install
```

### 6.33.2. Grep 软件包内容

安装的程序: egrep, fgrep, 和 grep

#### 简要介绍

egrep 打印匹配扩展正则表达式的行  
fgrep 打印匹配固定字符串列表的行  
grep 打印匹配基本正则表达式的行

## 6.34. Bash-5.0

The Bash package contains the Bourne-Again SHell.

大致构建用时: 1.7 SBU  
所需磁盘空间: 62 MB

### 6.34.1. Installation of Bash

Prepare Bash for compilation:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bash-5.0 \
            --without-bash-malloc \
            --with-installed-readline
```

The meaning of the new configure option:

*--with-installed-readline*

This option tells Bash to use the `readline` library that is already installed on the system rather than using its own `readline` version.

Compile the package:

```
make
```

Skip down to “Install the package” if not running the test suite.

To prepare the tests, ensure that the `nobody` user can write to the sources tree:

```
chown -Rv nobody .
```

Now, run the tests as the `nobody` user:

```
su nobody -s /bin/bash -c "PATH=$PATH HOME=/home make tests"
```

Install the package and move the main executable to `/bin`:

```
make install
mv -vf /usr/bin/bash /bin
```

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.

### 6.34.2. Contents of Bash

Installed programs: bash, bashbug, and sh (link to bash)  
Installed directory: /usr/share/doc/bash-5.0

#### 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

## 6.35. Libtool-2.4.6

Libtool 软件包包含 GNU 通用库支持脚本。它用一致的、可移植的接口封装复杂的共享库。

大致构建用时: 1.5 SBU  
所需磁盘空间: 43 MB

### 6.35.1. 安装 Libtool

准备编译 Libtool :

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

编译软件包 :

```
make
```

用以下命令测试结果(大约 11.0 SBU) :

```
make check
```

#### Note

在具有多个内核的系统上，libtool 的测试时间可以显著削减。为此，请在上面那行命令中添加 TESTSUITEFLAGS=-j<N>。例如，使用 -j4 或将减少 60% 的测试时间。

在 LFS 构建环境中已知有五个测试由于循环依赖会失败，但如果安装完 automake 之后重新检查，所有测试就都能通过。

安装软件包 :

```
make install
```

### 6.35.2. Libtool 软件包内容

安装的程序: libtool 和 libtoolize  
安装的库: libltdl.so  
安装目录: /usr/include/libltdl 和 /usr/share/libtool

#### 简要介绍

libtool	提供通用库编译支持服务
libtoolize	提供添加 libtool 支持到软件包的一个标准方法
libltdl	埋藏 dlopen 库的诸多难处

## 6.36. GDBM-1.18.1

The GDBM package contains the GNU Database Manager. It is a library of database functions that use extensible hashing and work similar to the standard UNIX dbm. The library provides primitives for storing key/data pairs, searching and retrieving the data by its key and deleting a key along with its data.

大致构建用时: 0.1 SBU  
所需磁盘空间: 11 MB

### 6.36.1. Installation of GDBM

Prepare GDBM for compilation:

```
./configure --prefix=/usr \
            --disable-static \
            --enable-libgdbm-compat
```

The meaning of the configure option:

`--enable-libgdbm-compat`

This switch enables the libgdbm compatibility library to be built, as some packages outside of LFS may require the older DBM routines it provides.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

### 6.36.2. Contents of GDBM

Installed programs: gdbm\_dump, gdbm\_load, and gdbmtool  
Installed libraries: libgdbm.so and libgdbm\_compat.so

#### Short Descriptions

<code>gdbm_dump</code>	Dumps a GDBM database to a file
<code>gdbm_load</code>	Recreates a GDBM database from a dump file
<code>gdbmtool</code>	Tests and modifies a GDBM database
<code>libgdbm</code>	Contains functions to manipulate a hashed database
<code>libgdbm_compat</code>	Compatibility library containing older DBM functions

## 6.37. Gperf-3.1

Gperf 为键集合生成完美的哈希函数。

大致构建用时:               少于 0.1 SBU

所需磁盘空间:               6.3 MB

### 6.37.1. 安装 Gperf

准备编译 Gperf :

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

编译软件包 :

```
make
```

该测试已知在运行多任务同时测试 (即 -j 选项大于 1) 时会失败。用以下命令测试结果 :

```
make -j1 check
```

安装软件包 :

```
make install
```

### 6.37.2. Gperf 软件包内容

安装的程序:               gperf

安装目录:                 /usr/share/doc/gperf-3.1

#### 简要介绍

gperf    为键集合生成完美哈希



## 6.38. Expat-2.2.6

Expat 软件包包含一个用于解析 XML 的面向流的 C 库。

大致构建用时: 0.1 SBU  
所需磁盘空间: 11 MB

### 6.38.1. 安装 Expat

首先修复一个 LFS 环境中，回归测试的问题：

```
sed -i 's|usr/bin/env |bin/|' run.sh.in
```

准备编译 Expat：

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.2.6
```

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

安装软件包：

```
make install
```

如果需要的话，安装帮助文档：

```
install -v -m644 doc/*.{html,png,css} /usr/share/doc/expat-2.2.6
```

### 6.38.2. Expat 软件包内容

安装的程序: xmlwf  
安装的库: libexpat.so  
安装目录: /usr/share/doc/expat-2.2.6

#### 简要介绍

xmlwf 用于检查 XML 文档是否格式良好的非验证工具  
libexpat 包含用于解析 XML 的 API 函数

## 6.39. Inetutils-1.9.4

Inetutils 软件包包含基本的网络程序。

大致构建用时: 0.3 SBU  
所需磁盘空间: 29 MB

### 6.39.1. 安装 Inetutils

准备编译 Inetutils :

```
./configure --prefix=/usr \
            --localstatedir=/var \
            --disable-logger \
            --disable-whois \
            --disable-rcp \
            --disable-rexec \
            --disable-rlogin \
            --disable-rsh \
            --disable-servers
```

配置选项的含义：

`--disable-logger`

该选项防止 Inetutils 安装 `logger` 程序，脚本使用该程序传递消息到系统日志守护进程。因为 Util-linux 安装了一个更新版本因此不能安装这个。

`--disable-whois`

该选项禁用编译过时的 Inetutils `whois` 客户端。BLFS 指南中有更好的 `whois` 客户端说明。

`--disable-r*`

为了安全，该参数使编译过时的程序不能被使用。提供该功能的程序在手册 BLFS 中的 `openssh` 会有所提及。

`--disable-servers`

禁用安装作为 Inetutils 软件包一部分的多种网络服务程序。这些服务程序被认为不适用于基础的 LFS 系统。其中的一些本来就不安全，或者说仅在可信网络中才被认为安全。注意这些服务程序有更好的可用替代品。

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

#### Note

测试 `libls.sh` 可能会在初始的 `chroot` 环境中失败，但是在 LFS 系统构建完成后重新运行就会通过了。测试 `ping-localhost.sh` 会因为宿主系统不支持 IPv6 而失败。

安装软件包：

```
make install
```

移动一些程序使得 `/usr` 在不可访问时仍保持可用：

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

### 6.39.2. Inetutils 软件包内容

安装的程序: `dnsdomainname`, `ftp`, `ifconfig`, `hostname`, `ping`, `ping6`, `talk`, `telnet`, `tftp`, 和 `traceroute`

#### 简要介绍

`dnsdomainname` 显示系统的 DNS 域名

ftp	简要介绍
hostname	报告或设置主机名称
ifconfig	管理网络接口
ping	发送请求应答包并报告响应用时
ping6	用于 IPv6 网络的 ping 版本
talk	用于和另一个用户交互
telnet	TELNET 协议接口
tftp	简单文件传输程序
traceroute	跟踪从你的工作主机发送到另一个网络上的主机的数据包通过的路径，显示中间通过的跳（网关）。

## 6.40. Perl-5.28.1

The Perl package contains the Practical Extraction and Report Language.

大致构建用时: 7.1 SBU  
所需磁盘空间: 274 MB

### 6.40.1. 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 test suite:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

This version of Perl now builds the `Compress::Raw::Zlib` and `Compress::Raw::BZip2` modules. By default Perl will use an internal copy of the sources for the build. Issue the following command so that Perl will use the libraries installed on the system:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

To have full control over the way Perl is set up, you can remove the “-des” options from the following command and hand-pick the way this package is built. Alternatively, use the command exactly as below to use the defaults that Perl auto-detects:

```
sh Configure -des -Dprefix=/usr \
               -Dvendorprefix=/usr \
               -Dman1dir=/usr/share/man/man1 \
               -Dman3dir=/usr/share/man/man3 \
               -Dpager="/usr/bin/less -isR" \
               -Duseshrplib \
               -Dusethreads
```

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 ensures that **less** is used instead of **more**.

`-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.

`-Duseshrplib`

Build a shared libperl needed by some perl modules.

`-Dusethreads`

Build perl with support for threads.

Compile the package:

```
make
```

To test the results (approximately 11 SBU), issue:

```
make -k test
```

#### Note

One test fails due to using the most recent version of gdbm.

Install the package and clean up:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

## 6.40.2. Contents of Perl

Installed programs:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.28.1 (hard link to perl), perlbug, perldoc, perlvp, perlthanks (hard link to perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp, and zipdetails
Installed libraries:	Many which cannot all be listed here
Installed directory:	/usr/lib/perl5

### Short Descriptions

corelist	A commandline frontend to Module::CoreList
cpan	Interact with the Comprehensive Perl Archive Network (CPAN) from the command line
enc2xs	Builds a Perl extension for the Encode module from either Unicode Character Mappings or Tcl Encoding Files
encguess	Guess the encoding type of one or several files
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 create a tarball from an installed module
json_pp	Converts data between certain input and output formats
libnetcfg	Can be used to configure the libnet Perl module
perl	Combines some of the best features of C, sed, awk and sh into a single swiss-army language
perl5.28.1	A hard link to perl
perlbug	Used to generate bug reports about Perl, or the modules that come with it, and mail them
perldoc	Displays a piece of documentation in pod format that is embedded in the Perl installation tree or in a Perl script
perlvp	The Perl Installation Verification Procedure; it can be used to verify that Perl and its libraries have been installed correctly
perlthanks	Used to generate thank you messages to mail to the Perl developers
piconv	A Perl version of the character encoding converter iconv
pl2pm	A rough tool for converting Perl4 .pl files to Perl5 .pm modules
pod2html	Converts files from pod format to HTML format
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
ptar	A tar-like program written in Perl
ptardiff	A Perl program that compares an extracted archive with an unextracted one
ptargrep	A Perl program that applies pattern matching to the contents of files in a tar archive
shasum	Prints or checks SHA checksums
splain	Is used to force verbose warning diagnostics in Perl
xsubpp	Converts Perl XS code into C code
zipdetails	Displays details about the internal structure of a Zip file

## 6.41. XML::Parser-2.44

XML::Parser 模块是到 James Clark 的 XML 解析器的 Perl Expat 接口。

大致构建用时:                少于 0.1 SBU

所需磁盘空间:               2.1 MB

### 6.41.1. 安装 XML::Parser

准备编译 XML::Parser :

```
perl Makefile.PL
```

编译软件包 :

```
make
```

用以下命令测试结果 :

```
make test
```

安装软件包 :

```
make install
```

### 6.41.2. XML::Parser 软件包内容

安装的模块:                Expat.so

#### 简要介绍

Expat 提供 Perl Expat 接口

## 6.42. Intltool-0.51.0

Intltool 是一个用于从源文件中抽取可翻译字符串的国际化工具。

大致构建用时:               少于 0.1 SBU  
所需磁盘空间:               1.5 MB

### 6.42.1. 安装 Intltool

首先修复 perl-5.22 和其后版本导致的警告：

```
sed -i 's:\\\\${:\\\\$\\{: intltool-update.in
```

准备编译 Intltool：

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

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

安装软件包：

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

### 6.42.2. Intltool 软件包内容

安装的程序:               intltool-extract, intltool-merge, intltool-prepare, intltool-update, 和 intltoolize  
安装的目录:               /usr/share/doc/intltool-0.51.0 and /usr/share/intltool

#### 简要介绍

intltoolize	准备使用 intltool 的软件包
intltool-extract	生成 gettext 能读取的头文件
intltool-merge	合并翻译后的字符串到多种文件格式
intltool-prepare	更新 pot 文件并把它和翻译文件合并
intltool-update	更新 po 模板文件并把它和翻译文件合并

## 6.43. Autoconf-2.69

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

大致构建用时:                少于 0.1 SBU (包含测试大于 2.7 SBU)  
所需磁盘空间:                17 MB

### 6.43.1. Installation of Autoconf

First, fix a bug generated by Perl 5.28.

```
sed '361 s/{/\\{/ -i bin/autoscan.in
```

Prepare Autoconf for compilation:

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

Compile the package:

```
make
```

The test suite is currently broken by bash-5 and libtool-2.4.3. To run the tests anyway, issue:

```
make check
```

Install the package:

```
make install
```

### 6.43.2. Contents of Autoconf

Installed programs:            autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, and ifnames  
Installed directory:           /usr/share/autoconf

#### Short Descriptions

autoconf	Produces shell scripts that automatically configure software source code packages to adapt to many kinds of Unix-like systems; the configuration scripts it produces are independent—running them does not require the autoconf program
autoheader	A tool for creating template files of C #define statements for configure to use
autom4te	A wrapper for the M4 macro processor
autoreconf	Automatically runs autoconf, autoheader, aclocal, automake, gettextize, and libtoolize in the correct order to save time when changes are made to autoconf and automake template files
autoscan	Helps to create a <code>configure.in</code> file for a software package; it examines the source files in a directory tree, searching them for common portability issues, and creates a <code>configure.scan</code> file that serves as a preliminary <code>configure.in</code> file for the package
autoupdate	Modifies a <code>configure.in</code> file that still calls autoconf macros by their old names to use the current macro names
ifnames	Helps when writing <code>configure.in</code> files for a software package; it prints the identifiers that the package uses in C preprocessor conditionals [If a package has already been set up to have some portability, this program can help determine what configure needs to check for. It can also fill in gaps in a <code>configure.in</code> file generated by autoscan.]



## 6.44. Automake-1.16.1

软件包包含了生成可与 Autoconf 一同使用的 Makefile 的程序。

大致构建用时: 少于 0.1 SBU (包含测试大于 6.9 SBU)

所需磁盘空间: 107 MB

### 6.44.1. Automake 的安装

准备编译 Automake :

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.1
```

编译软件包 :

```
make
```

因为各个测试之间存在内部延时，故建议就算是在单核处理器的设备上，也使用 -j4 编译选项加速测试过程。用以下命令测试结果：

```
make -j4 check
```

已知 LFS 环境中 subobj.sh 测试会失败。

安装软件包 :

```
make install
```

### 6.44.2. 关于 Automake 软件包内容

安装的程序: aclocal, aclocal-1.16 (到 aclocal 的硬链接), automake, 和 automake-1.16 (到 automake 的硬链接)

安装的目录: /usr/share/aclocal-1.16, /usr/share/automake-1.16, 和 /usr/share/doc/automake-1.16.1

#### 简要介绍

aclocal	基于 aclocal.m4 文件的内容生成 configure.in 文件
aclocal-1.16	到 aclocal 的硬链接
automake	一个从 Makefile.in 文件自动生成 Makefile.am 文件的工具 [要生成一个软件包里所有的 Makefile.in 文件，在最上层的目录运行这个程序。通过扫描 configure.in 文件，它能自动找到每个对应的 Makefile.am 文件，并生成对应的 Makefile.in 文件。]
automake-1.16	到 automake 的硬链接

## 6.45. Xz-5.2.4

Xz 软件包包含用于压缩和解压文件的程序。它提供 lzma 和更新的 xz 压缩格式功能。和传统的 gzip 或 bzip2 命令相比，用 xz 压缩文本文件能获得更好的压缩率。

大致构建用时：0.2 SBU  
所需磁盘空间：16 MB

### 6.45.1. 安装 Xz

准备编译 Xz：

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.4
```

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

安装软件包并确保所需的文件都在正确目录中：

```
make install
mv -v /usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat} /bin
mv -v /usr/lib/liblzma.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

### 6.45.2. Xz 软件包内容

安装的程序：lzcat (链接到 xz), lzcmp (链接到 xzdiff), lzdiff (链接到 xzdiff), lzegrep (链接到 xzgrep), lzfgrep (链接到 xzgrep), lzgrep (链接到 xzgrep), lzless (链接到 xzless), lzma (链接到 xz), lzmadec, lzmainfo, lzmore (链接到 xzmore), unlzma (链接到 xz), unxz (链接到 xz), xz, xzcat (链接到 xz), xzcmp (链接到 xzdiff), xzdec, xzdiff, xzegrep (链接到 xzgrep), xzfgrep (链接到 xzgrep), xzgrep, xzless, 和 xzmore  
安装的库：liblzma.so  
安装的目录：/usr/include/lzma 和 /usr/share/doc/xz-5.2.4

### 简要介绍

lzcat	解压标准输出
lzcmp	对 LZMA 压缩文件运行 cmp 命令
lzdiff	对 LZMA 压缩文件运行 diff 命令
lzegrep	对 LZMA 压缩文件运行 egrep 命令
lzfgrep	对 LZMA 压缩文件运行 fgrep 命令
lzgrep	对 LZMA 压缩文件运行 grep 命令
lzless	对 LZMA 压缩文件运行 less 命令
lzma	用 LZMA 格式压缩或解压文件
lzmadec	用于 LZMA 压缩文件的轻便解码器
lzmainfo	显示存储在 LZMA 压缩文件头部的信息
lzmore	对 LZMA 压缩文件运行 more 命令
unlzma	用 LZMA 格式解压文件
unxz	用 XZ 格式解压文件
xz	用 XZ 格式压缩或解压文件
xzcat	解压到标准输出
xzcmp	对 XZ 压缩文件运行 cmp 命令
xzdec	用于 XZ 压缩文件的轻便解码器

xzdiff	对 XZ 压缩文件运行 diff 命令
xzegrep	对 XZ 压缩文件运行 egrep 命令
xzfgrep	对 XZ 压缩文件运行 fgrep 命令
xzgrep	对 XZ 压缩文件运行 grep 命令
xzless	对 XZ 压缩文件运行 less 命令
xzmore	对 XZ 压缩文件运行 more 命令
liblzma	用 Lempel-Ziv-Markov 链算法实现无损块排序数据压缩的库

## 6.46. Kmod-26

Kmod 软件包包含用于加载内核模块的库和工具

大致构建用时: 0.1 SBU

所需磁盘空间: 13 MB

### 6.46.1. 安装 Kmod

准备编译 Kmod :

```
./configure --prefix=/usr \
            --bindir=/bin \
            --sysconfdir=/etc \
            --with-rootlibdir=/lib \
            --with-xz \
            --with-zlib
```

配置选项的含义 :

`--with-xz, --with-zlib`

这些选项使 Kmod 能处理压缩的内核模块。

`--with-rootlibdir=/lib`

该选项确保和不同库相关的文件放置到正确的目录。

编译软件包 :

```
make
```

这个软件包没有附带可在 LFS chroot 环境中运行测试套件。至少需要 git 程序并进行一些测试保证不会在 git 仓库外运行。

安装软件包并创建符号链接使兼容 Module-Init-Tools(之前处理 Linux 内核模块的软件包) :

```
make install

for target in depmod insmod lsmod modinfo modprobe rmmod; do
    ln -sfv ../bin/kmod /sbin/$target
done

ln -sfv kmod /bin/lsmod
```

### 6.46.2. Kmod 软件包内容

安装的程序: depmod (链接到 kmod), insmod (链接到 kmod), kmod, lsmod (链接到 kmod), modinfo (链接到 kmod), modprobe (链接到 kmod), 和 rmmod (链接到 kmod)

安装的库: libkmod.so

#### 简要介绍

depmod	基于从已有的模块集上发现的符号创建依赖文件 ; modprobe 用该依赖文件自动加载所需模块
insmod	在运行的内核上安装可加载模块
kmod	加载或卸载内核模块
lsmod	列出当前已加载模块
modinfo	检查和内核模块相关联的目标文件并显示搜索到的任何信息
modprobe	用 depmod 创建的依赖文件自动加载相关模块
rmmod	从运行中的内核卸载模块
libkmod	其它程序使用该库加载或卸载内核模块

## 6.47. Gettext-0.19.8.1

Gettext 软件包包含用于国际化和地区化的工具。这允许用 NLS (Native Language Support, 本地语言支持) 编译程序, 使得能以用户的本地语言输出信息。

大致构建用时: 2.0 SBU  
所需磁盘空间: 210 MB

### 6.47.1. 安装 Gettext

首先, 抑制两个 test-lock 的调用在某些机器上发生无限循环的问题:

```
sed -i '/^TESTS =/d' gettext-runtime/tests/Makefile.in &&
sed -i 's/test-lock..EXEEXT.//' gettext-tools/gnulib-tests/Makefile.in
```

然后修正配置文件:

```
sed -e '/AppData/{N;N;p;s/\.appdata\./.metainfo./}' \
-i gettext-tools/its/appdata.loc
```

准备编译 Gettext:

```
./configure --prefix=/usr \
--disable-static \
--docdir=/usr/share/doc/gettext-0.19.8.1
```

编译软件包:

```
make
```

用以下命令测试结果 (需要较长一段时间, 大概 3 SBUs):

```
make check
```

安装软件包:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

### 6.47.2. Gettext 软件包内容

安装的程序:	autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, 和 xgettext
安装的库:	libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, 和 preloadable_libintl.so
安装的目录:	/usr/lib/gettext, /usr/share/doc/gettext-0.19.8.1, 和 /usr/share/gettext

#### 简要介绍

autopoint	复制标准 Gettext 基础文件到源码包
envsubst	shell 格式字符串的替代环境变量
gettext	通过查看信息目录中的转换将原来语言信息转换为用户语言
gettext.sh	主要作为 gettext 的一个 shell 函数库
gettextize	复制所有标准 Gettext 文件到指定软件包的顶层目录以开始国际化
msgattrib	根据属性过滤翻译目录的信息并操作属性
msgcat	连接和合并给定 .po 文件
msgcmp	比较两个 .po 文件以检查两者是否包含相同的 msgid 字符串集合
msgcomm	查找给定 .po 文件共同的信息
msgconv	转换翻译目录到不同的字符编码
msgen	创建一个英语翻译目录
msgexec	对翻译目录的所有翻译运行命令
msgfilter	对翻译目录的所有翻译应用过滤器

<code>msgfmt</code>	从翻译目录生成一个二进制信息目录
<code>msggrep</code>	从翻译目录中抽取所有符合指定模式或属于特定源文件的信息
<code>msginit</code>	创建一个新的 <code>.po</code> 文件，根据用户环境中的值初始化元信息
<code>msgmerge</code>	合并两个原译到一个单独的文件
<code>msgunfmt</code>	反向编译一个二进制信息目录为原译文本
<code>msguniq</code>	统一重复翻译为一个翻译目录
<code>ngettext</code>	显示语法形式取决于多种母语翻译的文本信息的本地语言翻译
<code>recode-sr-latin</code>	对 Serbian 文本从 Cyrillic 重新编码为 Latin 脚本
<code>xgettext</code>	从指定源文件中抽取可翻译信息行用于生成第一个翻译模板
<code>libasprintf</code>	定义 <code>asprintf</code> 类，该类使 C 格式化输出在 C++ 程序中能和 <code>&lt;string&gt;</code> 字符串以及 <code>&lt;iostream&gt;</code> 流一起使用
<code>libgettextlib</code>	包括多种 Gettext 程序使用的常用例程的私有库；并不用于一般用途
<code>libgettextpo</code>	用来写处理 <code>.po</code> 文件的特殊程序；当 Gettext 附带的标准应用（例如 <code>msgcomm</code> ， <code>msgcmp</code> ， <code>msgattrib</code> ，以及 <code>msgen</code> ）不足够时会使用这个库
<code>libgettextsrc</code>	包括多种 Gettext 程序使用的常用例程的私有库；并不用于一般用途
<code>preloadable_libintl</code>	<code>LD_PRELOAD</code> 帮助 <code>libintl</code> 记录未翻译信息时使用的库

## 6.48. Libelf from Elfutils-0.176

Libelf is a library for handling ELF (Executable and Linkable Format) files.

大致构建用时: 1.3 SBU  
所需磁盘空间: 105 MB

### 6.48.1. Installation of Libelf

Libelf is part of elfutils-0.176 package. Use the elfutils-0.176.tar.bz2 as the source tarball.

Prepare Libelf for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install only Libelf:

```
make -C libelf install  
install -vm644 config/libelf.pc /usr/lib/pkgconfig
```

### 6.48.2. Contents of Libelf

Installed Library: libelf.so

## 6.49. Libffi-3.2.1

The Libffi library provides a portable, high level programming interface to various calling conventions. This allows a programmer to call any function specified by a call interface description at run time.

大致构建用时: 0.3 SBU  
所需磁盘空间: 7.6 MB

### 6.49.1. Installation of Libffi

#### Note

Similar to GMP, libffi builds with optimizations specific to the processor in use. If building for another system, export CFLAGS and CXXFLAGS to specify a generic build for your architecture. If this is not done, all applications that link to libffi will trigger Illegal Operation Errors.

Modify the Makefile to install headers into the standard `/usr/include` directory instead of `/usr/lib/libffi-3.2.1/include`.

```
sed -e '/^includesdir/ s/${libdir}.*$/$(includedir)/' \
    -i include/Makefile.in

sed -e '/^includedir/ s/=.*$/=@includedir/' \
    -e 's/^Cflags: -I${includedir}/Cflags:/' \
    -i libffi.pc.in
```

Prepare libffi for compilation:

```
./configure --prefix=/usr --disable-static --with-gcc-arch=native
```

The meaning of the configure option:

*--with-gcc-arch=native*

Ensure gcc optimizes for the current system. If this is not specified, the system is guessed and the code generated may not be correct for some systems. If the generated code will be copied from the native system to a less capable system, use the less capable system as a parameter. For details about alternative system types, see the x86 options in the gcc manual.

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

### 6.49.2. Contents of Libffi

Installed library: libffi.so

#### Short Descriptions

`libffi` contains the libffi API functions.



## 6.50. OpenSSL-1.1.1a

The OpenSSL package contains management tools and libraries relating to cryptography. These are useful for providing cryptographic functions to other packages, such as OpenSSH, email applications and web browsers (for accessing HTTPS sites).

大致构建用时: 1.7 SBU  
所需磁盘空间: 141 MB

### 6.50.1. Installation of OpenSSL

Prepare OpenSSL for compilation:

```
./config --prefix=/usr \
        --openssldir=/etc/ssl \
        --libdir=lib \
        shared \
        zlib-dynamic
```

Compile the package:

```
make
```

To test the results, issue:

```
make test
```

Install the package:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a/' Makefile
make MANSUFFIX=ssl install
```

If desired, install the documentation:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-1.1.1a
cp -vfr doc/* /usr/share/doc/openssl-1.1.1a
```

### 6.50.2. Contents of OpenSSL

Installed programs: c\_rehash and openssl  
Installed libraries: libcrypto.{so,a} and libssl.{so,a}  
Installed directories: /etc/ssl, /usr/include/openssl, /usr/lib/engines and /usr/share/doc/openssl-1.1.1a

### Short Descriptions

c_rehash	is a Perl script that scans all files in a directory and adds symbolic links to their hash values.
openssl	is a command-line tool for using the various cryptography functions of OpenSSL's crypto library from the shell. It can be used for various functions which are documented in man 1 openssl.
libcrypto.so	implements a wide range of cryptographic algorithms used in various Internet standards. The services provided by this library are used by the OpenSSL implementations of SSL, TLS and S/MIME, and they have also been used to implement OpenSSH, OpenPGP, and other cryptographic standards.
libssl.so	implements the Transport Layer Security (TLS v1) protocol. It provides a rich API, documentation on which can be found by running man 3 ssl.

## 6.51. Python-3.7.2

软件包 Python 3 包含了 Python 的开发环境。对于面向对象编程，书写脚本，构建大型程序的原型，或者开发整个应用程序而言，非常有用。

大致构建用时: 1.0 SBU  
所需磁盘空间: 392 MB

### 6.51.1. 安装 Python 3

编译 Python 前的准备：

```
./configure --prefix=/usr \
            --enable-shared \
            --with-system-expat \
            --with-system-ffi \
            --with-ensurepip=yes
```

配置选项的含义：

`--with-system-expat`  
该参数用于启用 Expat 系统版本的链接。

`--with-system-ffi`  
该参数用于启用 libffi 系统版本的链接。

`--with-ensurepip=yes`  
该参数用于启用 pip 和 setuptools 打包程序的构建。

编译软件包：

```
make
```

测试套件需要 TK 和 X Windows 会话，直至 BLFS 中重新安装 Python 3 之前都执行不了。

安装软件包：

```
make install
chmod -v 755 /usr/lib/libpython3.7m.so
chmod -v 755 /usr/lib/libpython3.so
```

安装命令的含义：

`chmod -v 755 /usr/lib/libpython3.{7m,}so`  
修复库的权限问题，同其他库保持一致。

如果需要，安装预格式化好的文档：

```
install -v -dm755 /usr/share/doc/python-3.7.2/html

tar --strip-components=1 \
    --no-same-owner \
    --no-same-permissions \
    -C /usr/share/doc/python-3.7.2/html \
    -xvf ../python-3.7.2-docs-html.tar.bz2
```

文档安装命令的含义：

`--no-same-owner` and `--no-same-permissions`  
确保安装文件的归属和权限是正确的。没有这个选项的话，运行 tar 时会以上游创建者的身份安装软件包内的文件。

### 6.51.2. Python 3 软件包内容

安装的程序: 2to3, idle3, pydoc3, python3, python3-config, pyvenv  
安装的库: libpython3.7m.so 和 libpython3.so  
安装的目录: /usr/include/python3.7m, /usr/lib/python3 和 /usr/share/doc/python-3.7.2

#### 简要介绍

2to3 一个用于读取 Python 2.x 源代码并实施一系列的修复，将其转化称有效的 Python 3.x 代码的 Python 程序。

idle3	一个用于打开 Python 自带的 GUI 编辑器的封装脚本。为了让该脚本能运行，你必须在安装 Python 前先安装 Tk，这样 Tkinter Python 模块才会构建。
pydoc3	Python 的文档工具。
python3	一种解释性的，交互式的，面向对象的编程语言。
pyenv	在一个或多个目标目录中，创建虚拟 Python 环境。

## 6.52. Ninja-1.9.0

Ninja is a small build system with a focus on speed.

大致构建用时: 0.2 SBU  
所需磁盘空间: 65 MB

### 6.52.1. Installation of Ninja

When run, ninja normally runs a maximum number of processes in parallel. By default this is the number of cores on the system plus two. In some cases this can overheat a CPU or run a system out of memory. If run from the command line, passing a `-jN` parameter will limit the number of parallel processes, but some packages embed the execution of ninja and do not pass a `-j` parameter.

Using the optional procedure below allows a user to limit the number of parallel processes via an environment variable, `NINJAJOBS`. For example, setting:

```
export NINJAJOBS=4
```

will limit ninja to four parallel processes.

If desired, add the capability to use the environment variable `NINJAJOBS` by running:

```
sed -i '/int Guess/a \
int    j = 0;\
char* jobs = getenv( "NINJAJOBS" );\
if ( jobs != NULL ) j = atoi( jobs );\
if ( j > 0 ) return j;\
' src/ninja.cc
```

Build Ninja with:

```
python3 configure.py --bootstrap
```

The meaning of the build option:

`--bootstrap`

This parameter forces ninja to rebuild itself for the current system.

To test the results, issue:

```
python3 configure.py
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

Install the package:

```
install -vm755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninja
install -vDm644 misc/zsh-completion /usr/share/zsh/site-functions/_ninja
```

### 6.52.2. Contents of Ninja

Installed programs:       ninja

#### Short Descriptions

ninja     is the Ninja build system.

## 6.53. Meson-0.49.2

Meson 是一个开源代码构建系统，不仅速度非常快，而且更重要的是对用户极其友好。

大致构建用时：    少于 0.1 SBU  
所需磁盘空间：    24 MB

### 6.53.1. 安装 Meson

通过下列命令编译 Meson：

```
python3 setup.py build
```

这个软件包还没有测试套件。

安装软件包：

```
python3 setup.py install --root=dest
cp -rv dest/* /
```

安装参数的含义：

`--root=dest`

默认情况下执行命令 `python3 setup.py install` 会将所有文件（比如 man 手册）安装到 Python Eggs。指定根路径则 `setup.py` 会将这些文件安装到标准层次目录。然后只需要复制层次目录，文件即处于标准位置。

### 6.53.2. Meson 软件包内容

安装的程序：    meson, mesonconf, mesonintrospect, mesontest, and wraptool  
安装的目录：    /usr/lib/python3.7/site-packages/meson-0.49.2-py3.7.egg

#### 简要介绍

meson	高生产率构建系统
mesonconf	Meson 构建系统配置工具
mesonintrospect	Meson 构建系统信息提取工具
mesontest	Meson 构建系统测试工具
wraptool	源码依赖下载器

## 6.54. Coreutils-8.30

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

大致构建用时: 2.6 SBU  
所需磁盘空间: 190 MB

### 6.54.1. Installation of Coreutils

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-8.30-i18n-1.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.

Suppress a test which on some machines can loop forever:

```
sed -i '/test.lock/s/^/#/' gnulib-tests/gnulib.mk
```

Now prepare Coreutils for compilation:

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
    --prefix=/usr \
    --enable-no-install-program=kill,uptime
```

The meaning of the configure options:

**autoreconf**

This command updates generated configuration files consistent with the latest version of automake.

**FORCE\_UNSAFE\_CONFIGURE=1**

This environment variable allows the package to be built as the root user.

**--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:

```
FORCE_UNSAFE_CONFIGURE=1 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 .
```

Now run the tests. Make sure the **PATH** in the **su** environment includes **/tools/bin**.

```
su nobody -s /bin/bash \
    -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

The test program **test-getlogin** is known to fail in a partially built system environment like the **chroot** environment here, but passes if run at the end of this chapter. The test program **tty.sh** is also known to fail.

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,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stat,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i s/"1"/"8"/1 /usr/share/man/man8/chroot.8
```

```
mv -v /usr/bin/{head,nice,sleep,touch} /bin
```

## 6.54.2. Contents of Coreutils

Installed programs: [, base32, base64, basename, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, and yes

Installed library: libstdbuf.so

Installed directory: /usr/libexec/coreutils

## Short Descriptions

base32	Encodes and decodes data according to the base32 specification (RFC 4648)
base64	Encodes and decodes data according to the base64 specification (RFC 4648)
basename	Strips any path and a given suffix from a file name
cat	Concatenates files to standard output
chcon	Changes security context for files and directories
chgrp	Changes the group ownership of files and directories
chmod	Changes the permissions of each file to the given mode; the mode can be either a symbolic representation of the changes to make or an octal number representing the new permissions
chown	Changes the user and/or group ownership of files and directories
chroot	Runs a command with the specified directory as the / directory
cksum	Prints the Cyclic Redundancy Check (CRC) checksum and the byte counts of each specified file
comm	Compares two sorted files, outputting in three columns the lines that are unique and the lines that are common
cp	Copies files
csplit	Splits a given file into several new files, separating them according to given patterns or line numbers and outputting the byte count of each new file
cut	Prints sections of lines, selecting the parts according to given fields or positions
date	Displays the current time in the given format, or sets the system date
dd	Copies a file using the given block size and count, while optionally performing conversions on it
df	Reports the amount of disk space available (and used) on all mounted file systems, or only on the file systems holding the selected files
dir	Lists the contents of each given directory (the same as the ls command)
dircolors	Outputs commands to set the LS_COLOR environment variable to change the color scheme used by ls

<code>dirname</code>	Strips the non-directory suffix from a file name
<code>du</code>	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
<code>echo</code>	Displays the given strings
<code>env</code>	Runs a command in a modified environment
<code>expand</code>	Converts tabs to spaces
<code>expr</code>	Evaluates expressions
<code>factor</code>	Prints the prime factors of all specified integer numbers
<code>false</code>	Does nothing, unsuccessfully; it always exits with a status code indicating failure
<code>fmt</code>	Reformats the paragraphs in the given files
<code>fold</code>	Wraps the lines in the given files
<code>groups</code>	Reports a user's group memberships
<code>head</code>	Prints the first ten lines (or the given number of lines) of each given file
<code>hostid</code>	Reports the numeric identifier (in hexadecimal) of the host
<code>id</code>	Reports the effective user ID, group ID, and group memberships of the current user or specified user
<code>install</code>	Copies files while setting their permission modes and, if possible, their owner and group
<code>join</code>	Joins the lines that have identical join fields from two separate files
<code>link</code>	Creates a hard link with the given name to a file
<code>ln</code>	Makes hard links or soft (symbolic) links between files
<code>logname</code>	Reports the current user's login name
<code>ls</code>	Lists the contents of each given directory
<code>md5sum</code>	Reports or checks Message Digest 5 (MD5) checksums
<code>mkdir</code>	Creates directories with the given names
<code>mkfifo</code>	Creates First-In, First-Outs (FIFOs), a "named pipe" in UNIX parlance, with the given names
<code>mknod</code>	Creates device nodes with the given names; a device node is a character special file, a block special file, or a FIFO
<code>mktemp</code>	Creates temporary files in a secure manner; it is used in scripts
<code>mv</code>	Moves or renames files or directories
<code>nice</code>	Runs a program with modified scheduling priority
<code>nl</code>	Numbers the lines from the given files
<code>nohup</code>	Runs a command immune to hangups, with its output redirected to a log file
<code>nproc</code>	Prints the number of processing units available to a process
<code>numfmt</code>	Converts numbers to or from human-readable strings
<code>od</code>	Dumps files in octal and other formats
<code>paste</code>	Merges the given files, joining sequentially corresponding lines side by side, separated by tab characters
<code>pathchk</code>	Checks if file names are valid or portable
<code>pinky</code>	Is a lightweight finger client; it reports some information about the given users
<code>pr</code>	Paginates and columnates files for printing
<code>printenv</code>	Prints the environment
<code>printf</code>	Prints the given arguments according to the given format, much like the C <code>printf</code> function
<code>ptx</code>	Produces a permuted index from the contents of the given files, with each keyword in its context
<code>pwd</code>	Reports the name of the current working directory
<code>readlink</code>	Reports the value of the given symbolic link
<code>realpath</code>	Prints the resolved path



<code>rm</code>	Removes files or directories
<code>rmdir</code>	Removes directories if they are empty
<code>runcon</code>	Runs a command with specified security context
<code>seq</code>	Prints a sequence of numbers within a given range and with a given increment
<code>sha1sum</code>	Prints or checks 160-bit Secure Hash Algorithm 1 (SHA1) checksums
<code>sha224sum</code>	Prints or checks 224-bit Secure Hash Algorithm checksums
<code>sha256sum</code>	Prints or checks 256-bit Secure Hash Algorithm checksums
<code>sha384sum</code>	Prints or checks 384-bit Secure Hash Algorithm checksums
<code>sha512sum</code>	Prints or checks 512-bit Secure Hash Algorithm checksums
<code>shred</code>	Overwrites the given files repeatedly with complex patterns, making it difficult to recover the data
<code>shuf</code>	Shuffles lines of text
<code>sleep</code>	Pauses for the given amount of time
<code>sort</code>	Sorts the lines from the given files
<code>split</code>	Splits the given file into pieces, by size or by number of lines
<code>stat</code>	Displays file or filesystem status
<code>stdbuf</code>	Runs commands with altered buffering operations for its standard streams
<code>stty</code>	Sets or reports terminal line settings
<code>sum</code>	Prints checksum and block counts for each given file
<code>sync</code>	Flushes file system buffers; it forces changed blocks to disk and updates the super block
<code>tac</code>	Concatenates the given files in reverse
<code>tail</code>	Prints the last ten lines (or the given number of lines) of each given file
<code>tee</code>	Reads from standard input while writing both to standard output and to the given files
<code>test</code>	Compares values and checks file types
<code>timeout</code>	Runs a command with a time limit
<code>touch</code>	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
<code>tr</code>	Translates, squeezes, and deletes the given characters from standard input
<code>true</code>	Does nothing, successfully; it always exits with a status code indicating success
<code>truncate</code>	Shrinks or expands a file to the specified size
<code>tsort</code>	Performs a topological sort; it writes a completely ordered list according to the partial ordering in a given file
<code>tty</code>	Reports the file name of the terminal connected to standard input
<code>uname</code>	Reports system information
<code>unexpand</code>	Converts spaces to tabs
<code>uniq</code>	Discards all but one of successive identical lines
<code>unlink</code>	Removes the given file
<code>users</code>	Reports the names of the users currently logged on
<code>vdir</code>	Is the same as <code>ls -l</code>
<code>wc</code>	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
<code>who</code>	Reports who is logged on
<code>whoami</code>	Reports the user name associated with the current effective user ID
<code>yes</code>	Repeatedly outputs “y” or a given string until killed
<code>libstdbuf</code>	Library used by <code>stdbuf</code>

## 6.55. Check-0.12.0

Check 是 C 语言的单元测试框架。

大致构建用时: 0.1 SBU (包含测试大于 3.0 SBU)  
所需磁盘空间: 12 MB

### 6.55.1. 安装 Check

配置 Check 准备编译：

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

构建软件包：

```
make
```

现在编译完成了。运行 Check 的测试套件，输入以下命令：

```
make check
```

注意，Check 的测试套件可能会占用挺长（上至 4 SBU）的时间。

安装软件包，并修复脚本：

```
make install  
sed -i '1 s/tools/usr/' /usr/bin/checkmk
```

### 6.55.2. Check 软件包内容

安装的程序: checkmk  
安装的库: libcheck.{a,so}

#### 简要介绍

checkmk 用于生成 C 语言单元测试的 Awk 脚本，这些用例可以配合 Check 单元测试框架使用  
libcheck.{a,so} 包含允许测试程序调用 Check 的功能

## 6.56. Diffutils-3.7

Diffutils 软件包包含显示文件和目录差异的程序。

大致构建用时: 0.3 SBU  
所需磁盘空间: 36 MB

### 6.56.1. 安装 Diffutils

准备编译 Diffutils :

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

编译软件包 :

```
make
```

用以下命令测试结果 :

```
make check
```

安装软件包 :

```
make install
```

### 6.56.2. Diffutils 软件包内容

安装的程序: cmp, diff, diff3, and sdiff

#### 简要介绍

cmp	比较两个文件并报告字节差异
diff	比较两个文件或目录并报告文件中的行差异
diff3	逐行比较三个文件
sdiff	比较两个文件并交互式输出结果

## 6.57. Gawk-4.2.1

The Gawk package contains programs for manipulating text files.

大致构建用时: 0.3 SBU  
所需磁盘空间: 44 MB

### 6.57.1. Installation of Gawk

First, ensure some unneeded files are not installed:

```
sed -i 's/extras//' Makefile.in
```

Prepare Gawk for compilation:

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

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-4.2.1
cp -v doc/{awkforai.txt,*.eps,pdf,jpg} /usr/share/doc/gawk-4.2.1
```

### 6.57.2. Contents of Gawk

Installed programs:	awk (link to gawk), gawk, and awk-4.2.1
Installed libraries:	filefuncs.so, fnmatch.so, fork.so, inplace.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rvarray.so, testtext.so, and time.so
Installed directories:	/usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, and /usr/share/doc/gawk-4.2.1

### Short Descriptions

awk	A link to gawk
gawk	A program for manipulating text files; it is the GNU implementation of awk
gawk-4.2.1	A hard link to gawk

## 6.58. Findutils-4.6.0

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).

大致构建用时: 0.6 SBU  
所需磁盘空间: 51 MB

### 6.58.1. Installation of Findutils

First, suppress a test which on some machines can loop forever:

```
sed -i 's/test-lock..EXEXT.//' tests/Makefile.in
```

Next, make some fixes required by glibc-2.28 and later:

```
sed -i 's/IO_ftrylockfile/IO_EOF_SEEN/' gl/lib/*.c
sed -i '/unistd/a #include <sys/sysmacros.h>' gl/lib/mountlist.c
echo "#define _IO_IN_BACKUP 0x100" >> gl/lib/stdio-impl.h
```

Prepare Findutils for compilation:

```
./configure --prefix=/usr --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 packages in BLFS and beyond expect the find program in `/bin`, so make sure it's placed there:

```
mv -v /usr/bin/find /bin
sed -i 's|find:=${BINDIR}|find:=/bin|' /usr/bin/updatedb
```

### 6.58.2. Contents of Findutils

Installed programs: find, locate, updatedb, and xargs

#### Short Descriptions

find	Searches given directory trees for files matching the specified criteria
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

## 6.59. Groff-1.22.4

Groff 软件包包含用于处理和格式化文本的程序。

大致构建用时: 0.4 SBU  
所需磁盘空间: 94 MB

### 6.59.1. 安装 Groff

Groff 希望环境变量 `PAGE` 包含默认的页面大小, 对于美国的用户, 为 `PAGE=letter`, 对于其它地方, `PAGE=A4` 更合适。尽管在编译的时候配置了默认页面大小, 后面通过 `echo` 「A4」或「letter」到 `/etc/papersize` 文件仍然可以修改。

准备编译 Groff :

```
PAGE=<paper_size> ./configure --prefix=/usr
```

该软件不支持并行构建。编译软件包 :

```
make -j1
```

该软件包没有测试套具。

安装软件包 :

```
make install
```

### 6.59.2. Groff 软件包内容

安装的程序: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, 和 troff

安装的目录: /usr/lib/groff 和 /usr/share/doc/groff-1.22.4, /usr/share/groff

#### 简要介绍

addftinfo	读 troff 字体文件并添加一些额外的 groff 系统使用的字体信息
afmtodit	创建用于和 groff 以及 grops 一起使用的字体文件
chem	Groff 预处理生成化学结构图
eqn	编译嵌入了 troff 输入文件的方程的描述为 troff 能理解的命令
eqn2graph	转换 troff EQN (equation 方程) 为裁剪图像
gdiffmk	标记 groff/nroff/troff 文件的差异
glilypond	将 lilypond 语言写成的乐谱转换至 groff语言
gperl	Groff 的预处理器, 允许在 groff 中添加 perl 代码
gpinyin	Groff 的预处理器, 允许在 groff 文件中添加类似欧洲语言的中文拼音。
grap2graph	转换 grap 图为裁剪位图图像
grn	用于 gremlin 文件的 groff 预处理器
grodvi	生成 TeX dvi 格式的 groff 驱动
groff	groff 文档格式化系统前端; 一般运行 troff 程序和适合选定文件的后处理器
groffer	在 X 和 tty 终端显示 groff 文件以及 man 页面
grog	读文件并猜测打印文件需要的 groff 选项 <code>-e</code> , <code>-man</code> , <code>-me</code> , <code>-mm</code> , <code>-ms</code> , <code>-p</code> , <code>-s</code> , 以及 <code>-t</code> , 并报告包含这些选项的 groff 命令
grolbp	用于 Canon CAPSL 打印机 (LBP-4 和 LBP-8 系列激光打印机) 的 groff 驱动
grolj4	生成适合于 HP LaserJet 4 打印机的 PCL5 格式输出的 groff 驱动
gropdf	翻译 GNU troff 的输出至 PDF
grops	转换 GNU troff 输出为 PostScript
grotty	转换 GNU troff 输出为适合于打字机设备的格式

hpftodit	从 HP 标签字体规格文件创建和 groff -Tlj4 一起使用的字体文件
indxbib	用和 refer, lookbib, 以及 lkbib 一起使用的指定文件为文献数据库创建倒排索引
lkbib	搜索文献数据库中指定键的引用并报告找到的任何引用
lookbib	在标准错误中输出提示（除非标准输入不是终端），从标准输入读取包含一系列关键字的行，在指定的文件中搜索文献数据库中那些关键字的引用，在标准输出中打印找到的任何引用，循环这些过程直到输入结束
mmroff	groff 的简单预处理器
neqn	为美国标准信息交换码 (ASCII) 输出格式化方程
nroff	用 nroff 模仿 groff 命令的脚本
pdfmom	groff 周围的封装，用于简化从带有 mom 宏的文件格式生成 PDF 文件的过程。
pdfroff	用 groff 创建 pdf 文档
pfbtops	转换 .pfb 格式中的 PostScript 字体为 ASCII
pic	编译嵌入了 troff 或 TeX 输入文件的图像的描述为 TeX 或 troff 能理解的命令
pic2graph	转换 PIC 图为裁剪图像
post-grohtml	转换 GNU troff 输出为 HTML
preconv	转换输入文件编码为 GNU troff 能理解的编码
pre-grohtml	转换 GNU troff 的输出为 HTML
refer	复制文件内容到标准输出，其中 .[ 和 .] 之间的行解释为引用，.R1 和 .R2 之间的行解释为如何处理引用的命令
roff2dvi	转化 roff 文件到 DVI 格式
roff2html	转换 roff 文件到 HTML 格式
roff2pdf	转换 roff 文件到 PDFs
roff2ps	转换 roff 文件为 ps 文件
roff2text	转换 roff 文件为文本文件
roff2x	转换 roff 文件到其它格式
soelim	读文件并用相应的 file 内容替换 .so file 格式的行
tbl	编译嵌入了 troff 输入文件的表的描述为 troff 能理解的命令
tfmtodit	创建和 groff -Tdvi 一起使用的字体文件
troff	和 Unix troff 高度兼容；通常应该使用 groff 命令调用，它也会以恰当的顺序和选项运行预处理器和后处理器

## 6.60. GRUB-2.02

The GRUB package contains the GRand Unified Bootloader.

大致构建用时: 0.6 SBU  
所需磁盘空间: 147 MB

### 6.60.1. Installation of GRUB

Prepare GRUB for compilation:

```
./configure --prefix=/usr \
            --sbindir=/sbin \
            --sysconfdir=/etc \
            --disable-efiemu \
            --disable-werror
```

The meaning of the new configure options:

*--disable-werror*

This allows the build to complete with warnings introduced by more recent Flex versions.

*--disable-efiemu*

This option minimizes what is built by disabling a feature and testing programs not needed for LFS.

Compile the package:

```
make
```

This package does not come with a test suite.

Install the package:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Using GRUB to make your LFS system bootable will be discussed in Section#8.4, “Using GRUB to Set Up the Boot Process”.

### 6.60.2. Contents of GRUB

Installed programs:	grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkreldpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup, and grub-syslinux2cfg
Installed directories:	/usr/lib/grub, /etc/grub.d, /usr/share/grub, and boot/grub (when grub-install is first run)

### Short Descriptions

grub-bios-setup	Is a helper program for grub-install
grub-editenv	A tool to edit the environment block
grub-file	Checks if FILE is of the specified type.
grub-fstest	Tool to debug the filesystem driver
grub-glue-efi	Processes ia32 and amd64 EFI images and glues them according to Apple format.
grub-install	Install GRUB on your drive
grub-kbdcomp	Script that converts an xkb layout into one recognized by GRUB
grub-macbless	Mac-style bless on HFS or HFS+ files
grub-menulst2cfg	Converts a GRUB Legacy menu.lst into a grub.cfg for use with GRUB 2
grub-mkconfig	Generate a grub config file
grub-mkimage	Make a bootable image of GRUB



<code>grub-mklayout</code>	Generates a GRUB keyboard layout file
<code>grub-mknetdir</code>	Prepares a GRUB netboot directory
<code>grub-mkpasswd-pbkdf2</code>	Generates an encrypted PBKDF2 password for use in the boot menu
<code>grub-mkrelpath</code>	Makes a system pathname relative to its root
<code>grub-mkrescue</code>	Make a bootable image of GRUB suitable for a floppy disk or CDROM/DVD
<code>grub-mkstandalone</code>	Generates a standalone image
<code>grub-ofpathname</code>	Is a helper program that prints the path of a GRUB device
<code>grub-probe</code>	Probe device information for a given path or device
<code>grub-reboot</code>	Sets the default boot entry for GRUB for the next boot only
<code>grub-render-label</code>	Render Apple .disk_label for Apple Macs
<code>grub-script-check</code>	Checks GRUB configuration script for syntax errors
<code>grub-set-default</code>	Sets the default boot entry for GRUB
<code>grub-sparc64-setup</code>	Is a helper program for grub-setup
<code>grub-syslinux2cfg</code>	Transform a syslinux config file into grub.cfg format

## 6.61. Less-530

Less 软件包包含一个文本文件查看器。

大致构建用时:           少于 0.1 SBU  
所需磁盘空间:           3.9 MB

### 6.61.1. 安装 Less

准备编译 Less :

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

配置选项的含义 :

`--sysconfdir=/etc`

该选项告诉软件包创建的程序在 `/etc` 中查找配置文件。

编译软件包 :

```
make
```

该软件包没有测试套件。

安装软件包 :

```
make install
```

### 6.61.2. Less 软件包内容

安装的程序:           less, lessecho, 和 lesskey

#### 简要介绍

less	文件查看器或分页器；显示指定文件的内容，允许用户滚动、查找字符串以及跳转到标记。
lessecho	在 Unix 系统文件名中扩展元字符，例如 * 和 ?
lesskey	用于指定绑定到 less 的键

## 6.62. Gzip-1.10

The Gzip package contains programs for compressing and decompressing files.

大致构建用时: 0.1 SBU  
所需磁盘空间: 20 MB

### 6.62.1. Installation of Gzip

Prepare Gzip for compilation:

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

Compile the package:

```
make
```

To test the results, issue:

```
make check
```

Two tests are known to fail in the LFS environment: help-version and zmore.

Install the package:

```
make install
```

Move a program that needs to be on the root filesystem:

```
mv -v /usr/bin/gzip /bin
```

### 6.62.2. Contents of Gzip

Installed programs: gunzip, gzexe, gzip, uncompress (hard link with gunzip), 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

## 6.63. IPRoute2-4.20.0

IPRoute2 软件包包含基于 IPV4 网络的基本和高级程序。

大致构建用时: 0.2 SBU  
所需磁盘空间: 13 MB

### 6.63.1. 安装 IPRoute2

此软件包中包含的 arpd 因为依赖于 Berkeley DB，但是此软件并没有包含于 LFS 中，所以将不会进行编译。但是 arpd 的目录依旧会被安装。运行以下的命令来阻止这一动作。如果需要 arpd 的二进制文件，请查看 BLFS Book 的网页 <http://www.linuxfromscratch.org/blfs/view/8.4/server/databases.html#db> 以了解编译 Berkeley DB 都需要哪些指令。

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

此外，还需要禁用两个模块，它依赖于 <http://www.linuxfromscratch.org/blfs/view/8.4/postlfs/iptables.html>。

```
sed -i 's/.m_ipt.o//' tc/Makefile
```

编译软件包：

```
make
```

此软件包不包含可用的测试套件。

安装软件包：

```
make DOCDIR=/usr/share/doc/iproute2-4.20.0 install
```

### 6.63.2. IPRoute2 软件包内容

安装的程序: bridge, ctstat (链接到 Instat), genl, ifcfg, ifstat, ip, Instat, nstat, route, routel, rtacct, rtmon, rtpr, rtstat (链接到 Instat), ss, 和 tc  
安装的目录: /etc/iproute2, /usr/lib/tc, 和 /usr/share/doc/iproute2-4.20.0,

### 简要介绍

bridge	配置网桥
ctstat	连接状态工具
genl	通用 netlink 多用途前端
ifcfg	ip 命令的 shell 脚本封装 [注意它需要 iputils 软件包中的 arping 和 rdisk 程序，可以在 <a href="http://www.skbuff.net/iputils/">http://www.skbuff.net/iputils/</a> 找到 iputils 软件包。]
ifstat	显示接口统计信息，包括接口发送和接收的包的数目
ip	主要的可执行程序。它有多种不同功能： ip link <device> 允许用户查看设备状态或更改 ip addr 允许用户查看地址和属性、增加新地址、删除旧地址 ip neighbor 允许用户查看邻居和它们的特性、增加新邻居、删除旧邻居 ip rule 允许用户查看路由策略并更改 ip route 允许用户查看路由表并更改路由表规则 ip tunnel 允许用户查看 IP 隧道及其特性、并进行更改 ip maddr 允许用户查看多播地址及其特性、并进行更改 ip mroute 允许用户设置、更改或删除多播路由 ip monitor 允许用户持续监视设置、地址和路由状态
Instat	提供 Linux 网络统计信息；是更通用、功能更完备的替代旧 rtstat 的程序
nstat	显示网络统计信息
route	ip route 的组件。用于清空路由表
routel	ip route 的组件。用于列出路由表
rtacct	显示 /proc/net/route 的内容
rtmon	路由监视工具

rtpr	转换 ip -o 输出为可读形式
rtstat	路由状态工具
ss	类似于 netstat 命令；显示活动连接
tc	拥塞控制可执行程序；用于实现服务质量 (Quality Of Service,QOS) 和服务等级 (Class Of Service,COS) tc qdisc 允许用户设置排队规则 tc class 允许用户基于排队规则调度策略设置等级 tc estimator 允许用户估计到一个网络的网络流量 tc filter 允许用户设置 QOS/COS 包过滤 tc policy 允许用户设置 QOS/COS 策略

## 6.64. Kbd-2.0.4

Kbd 软件包包含键表文件、控制台字体和键盘工具。

大致构建用时: 0.1 SBU  
所需磁盘空间: 30 MB

### 6.64.1. 安装 Kbd

在 Kbd 软件包中退格键 (Backspace) 和删除键 (Delete) 的行为和键映射并不一致。下面的补丁修复了 i386 键映射中的这个问题：

```
patch -Np1 -i ../kbd-2.0.4-backspace-1.patch
```

打补丁后，退格键生成编码为 127 的字符，删除键会生成一个著名的转义序列。

移除冗余的 `resizecons` 程序（它要求功能不全的 `svglib` 提供视频模式文件——用于正常使用 `setfont` 设置控制台字体大小）以及帮助手册。

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\lno/g' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

准备编译 Kbd：

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr --disable-vlock
```

配置选项的含义：

`--disable-vlock`

该选项防止编译 `vlock` 工具，因为它要求 `chroot` 环境中不可用的 PAM 库。

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

安装软件包：

```
make install
```

#### Note

由于通常使用的 CP1251 键映射假设使用 ISO-8859-5 编码，Kbd 软件包不能为某些语言（例如，白俄罗斯）提供可用的键映射。使用这样的语言需要单独下载能工作的键映射。

如果需要的话，安装帮助文档：

```
mkdir -v /usr/share/doc/kbd-2.0.4
cp -R -v docs/doc/* /usr/share/doc/kbd-2.0.4
```

### 6.64.2. Kbd 软件包内容

安装的程序: `chvt`, `deallocvt`, `dumpkeys`, `fgconsole`, `getkeycodes`, `kbdinfo`, `kbd_mode`, `kbdrate`, `loadkeys`, `loadunimap`, `mapscrm`, `openvt`, `psfaddtable` (链接到 `psfxtable`), `psfgettable` (链接到 `psfxtable`), `psfstriptime` (链接到 `psfxtable`), `psfxtable`, `setfont`, `setkeycodes`, `setleds`, `setmetamode`, `setvtrgb`, `showconsolefont`, `showkey`, `unicode_start`, 和 `unicode_stop`

安装目录: `/usr/share/consolefonts`, `/usr/share/consoletrans`, `/usr/share/keymaps`, `/usr/share/doc/kbd-2.0.4`, 和 `/usr/share/unimaps`

#### 简要介绍

`chvt` 更改前台虚拟终端  
`deallocvt` 重新分配未使用的虚拟终端

dumpkeys	转储键盘转换表
fgconsole	输出活动虚拟终端的数目
getkeycodes	输出内核扫描码到键码的映射表
kbdinfo	获取关于某个终端的状态信息
kbd_mode	报告或设置键盘模式
kbdrate	设置键盘重复和延迟速度
loadkeys	加载键盘转换表
loadunimap	加载内核 Unicode 到字体映射表
mapscrn	用于加载用户定义的输出字符映射表到控制台驱动过的过时程序；现在通过 setfont 完成
openvt	在一个新的虚拟终端 (VT) 启动程序
psfaddtable	添加 Unicode 字符表到控制台字体
psfgettable	从控制台字体抽取嵌入的 Unicode 字符表
psfstriptide	从控制台字体移除嵌入的 Unicode 字符表
psfxtable	为控制台字体处理 Unicode 字符表
setfont	更改控制台的增强图形适配器 (Enhanced Graphic Adapter,EGA) 和视频图形阵列 (Video Graphics Array,VGA) 的字体
setkeycodes	加载内核扫描码到键码映射表条目；键盘上有异常键时非常有用
setleds	设置键盘标记和 LED 灯
setmetamode	定义键盘元键处理
setvtrgb	设置所有虚拟终端中的控制台颜色映射
showconsolefont	显示当前 EGA/VGA 控制台屏幕字体
showkey	报告键盘上按键的扫描码、键码以及 ASCII 码
unicode_start	设置键盘和控制台为 UNICODE 模式[别用该程序，除非你的键映射文件是 ISO-8859-1 编码。对于其它编码，该工具会输出错误结果。]
unicode_stop	从 UNICODE 模式恢复键盘和控制台为原来模式

## 6.65. Libpipeline-1.5.1

Libpipeline 软件包包含以灵活方便方式管理管道和子进程的库。

大致构建用时: 0.1 SBU  
所需磁盘空间: 9.0 MB

### 6.65.1. 安装 Libpipeline

准备编译 Libpipeline :

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

编译软件包 :

```
make
```

用以下命令测试结果 :

```
make check
```

安装软件包 :

```
make install
```

### 6.65.2. Libpipeline 软件包内容

安装的库: libpipeline.so

#### 简要介绍

`libpipeline` 该库用于在子进程之间安全地建立管道



## 6.66. Make-4.2.1

The Make package contains a program for compiling packages.

大致构建用时: 0.5 SBU  
所需磁盘空间: 13 MB

### 6.66.1. Installation of Make

Again, work around an error caused by glibc-2.27 and later:

```
sed -i '211,217 d; 219,229 d; 232 d' glob/glob.c
```

Prepare Make for compilation:

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

Compile the package:

```
make
```

The test suite needs to know where supporting perl files are located. We use an environment variable to accomplish this. To test the results, issue:

```
make PERL5LIB=$PWD/tests/ check
```

Install the package:

```
make install
```

### 6.66.2. 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

## 6.67. Patch-2.7.6

Patch 软件包包含一个通过打「补丁」创建或修改文件的程序，补丁文件通常由 diff 程序生成。

大致构建用时: 0.2 SBU  
所需磁盘空间: 13 MB

### 6.67.1. 安装 Patch

准备编译 Patch :

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

编译软件包 :

```
make
```

用以下命令测试结果 :

```
make check
```

安装软件包 :

```
make install
```

### 6.67.2. Patch 软件包内容

安装的程序: patch

#### 简要介绍

patch 根据补丁文件修改文件[补丁文件通常是由 diff 程序产生的差异列表。通过对源文件应用这些差异生成打补丁后的版本。]

## 6.68. Man-DB-2.8.5

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

大致构建用时: 0.3 SBU  
所需磁盘空间: 36 MB

### 6.68.1. Installation of Man-DB

Prepare Man-DB for compilation:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/man-db-2.8.5 \
            --sysconfdir=/etc \
            --disable-setuid \
            --enable-cache-owner=bin \
            --with-browser=/usr/bin/lynx \
            --with-vgrind=/usr/bin/vgrind \
            --with-grap=/usr/bin/grap
```

The meaning of the configure options:

*--disable-setuid*

This disables making the man program setuid to user man.

*--enable-cache-owner=bin*

This makes the system-wide cache files be owned by user bin.

*--with-...*

These three parameters are used to set some default programs. 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
```

To test the results, issue:

```
make check
```

Install the package:

```
make install
```

### 6.68.2. Non-English Manual Pages in LFS

The following table shows the character set that Man-DB assumes manual pages installed under `/usr/share/man/<ll>` will be encoded with. In addition to this, Man-DB correctly determines if manual pages installed in that directory are UTF-8 encoded.

Table#6.1.#Expected character encoding of legacy 8-bit manual pages

Language (code)	Encoding	Language (code)	Encoding
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2

Language (code)	Encoding	Language (code)	Encoding
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7	#	#

## Note

Manual pages in languages not in the list are not supported.

## 6.68.3. Contents of Man-DB

Installed programs:      accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, and whatis  
 Installed libraries:     libman.so and libmandb.so  
 Installed directories:   /usr/lib/man-db, /usr/lib/tmpfiles.d, /usr/libexec/man-db, and /usr/share/doc/man-db-2.8.5

## Short Descriptions

**accessdb**      Dumps the whatis database contents in human-readable form  
**apropos**      Searches the whatis database and displays the short descriptions of system commands that contain a given string  
**catman**       Creates or updates the pre-formatted manual pages  
**lexgrog**       Displays one-line summary information about a given manual page  
**man**           Formats and displays the requested manual page  
**mandb**        Creates or updates the whatis database  
**manpath**      Displays the contents of \$MANPATH or (if \$MANPATH is not set) a suitable search path based on the settings in man.conf and the user's environment  
**whatis**        Searches the whatis database and displays the short descriptions of system commands that contain the given keyword as a separate word  
**libman**       Contains run-time support for man  
**libmandb**     Contains run-time support for man

## 6.69. Tar-1.31

The Tar package contains an archiving program.

大致构建用时: 1.7 SBU  
所需磁盘空间: 45 MB

### 6.69.1. Installation of Tar

Fix a bug introduced in tar-1.31:

```
sed -i 's/abort.*/FALLTHROUGH;/' src/extract.c
```

Prepare Tar for compilation:

```
FORCE_UNSAFE_CONFIGURE=1 \
./configure --prefix=/usr \
--bindir=/bin
```

The meaning of the configure options:

`FORCE_UNSAFE_CONFIGURE=1`

This forces the test for `mknod` to be run as root. It is generally considered dangerous to run this test as the root user, but as it is being run on a system that has only been partially built, overriding it is OK.

Compile the package:

```
make
```

To test the results (about 3 SBU), issue:

```
make check
```

Install the package:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.31
```

### 6.69.2. Contents of Tar

Installed programs: tar  
Installed directory: /usr/share/doc/tar-1.31

### Short Descriptions

**tar** Creates, extracts files from, and lists the contents of archives, also known as tarballs

## 6.70. Texinfo-6.5

Texinfo 软件包包含用于读、写以及转换信息页的程序。

大致构建用时: 0.9 SBU  
所需磁盘空间: 129 MB

### 6.70.1. 安装 Texinfo

修复在回归检测中导致大量失败项的文件：

```
sed -i '5481,5485 s/{/{/ tp/TeXinfo/Parser.pm
```

准备编译 Texinfo：

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

配置选项的含义：

`--disable-static`

顶级的配置脚本会告诉你这是一个未能识别的选项，但是 XSParagraph 的配置脚本能够识别它，并能用其来禁用安装静态 XSParagraph.a 至 /usr/lib/texinfo 的操作。

编译软件包：

```
make
```

用以下命令测试结果：

```
make check
```

安装软件包：

```
make install
```

可选地安装 TeX 中的组件：

```
make TEXMF=/usr/share/texmf install-tex
```

make 参数的含义：

`TEXMF=/usr/share/texmf`

如果后面会安装 TeX 软件包，TEXMF makefile 变量保存了作为 TeX 树的根位置。

该信息文档系统使用一个纯文本文件来存放菜单条目清单。文件保存在 /usr/share/info/dir。不幸的是，由于不同软件包 Makefile 的偶然问题，有时候会和系统中安装的信息页不同步。如果需要重建 /usr/share/info/dir 文件，下面的可选命令能完成任务：

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

### 6.70.2. Texinfo 软件包内容

安装的程序: info, install-info, makeinfo (链接到 texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, 和 texindex  
安装的库: XSParagraph.so  
安装的目录: /usr/share/texinfo 和 /usr/lib/texinfo

#### 简要介绍

info	用于读取和 man 页面相似的信息页，但和只是解释所有可用命令行选项相比更加深入，[例如比较 man bison 和 info bison。]
install-info	用于安装信息页；它会更新 info 索引文件中的条目
makeinfo	翻译给定的 Texinfo 源文档为信息页、纯文本或 HTML
pdftexi2dvi	用于格式化给定的 Texinfo 文档为 PDF 文件

pod2texi	转换 Pod 为 Texinfo 格式
texi2any	翻译 Texinfo 源文档为多种其它格式
texi2dvi	用于格式化给定 Texinfo 文档为可打印的设备无关文件
texi2pdf	用于格式化给定 Texinfo 文档为 PDF 文件
texindex	用于对 Texinfo 索引文件进行排序

## 6.71. Vim-8.1

Vim 软件包包含了一个强大的文本编辑器。

大致构建用时: 1.3 SBU

所需磁盘空间: 169 MB

### Vim 的安装

如果你钟情于其它的编辑器，比如 Emacs、Joe，或 Nano。请参考 <http://www.linuxfromscratch.org/blfs/view/8.4/postlfs/editors.html> 里的安装指导。

### 6.71.1. Vim 的安装

首先，把配置文件 `vimrc` 从默认位置移动到 `/etc`：

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

让 Vim 做好编译准备：

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

编译安装包：

```
make
```

检验结果：

```
LANG=en_US.UTF-8 make -j1 test &> vim-test.log
```

这个测试套件会输出一堆二进制数据到屏幕上。这会导致当前设置下的终端出现问题。把输出重定向到一个日志文件就可以解决这个问题。测试成功的话就会输出「ALL DONE」。

安装软件包：

```
make install
```

许多用户习惯于使用 `vi` 而不是 `vim`。为了当人们在习惯性的输入 `vi` 时能执行 `vim`，需要给二进制文件和 `man` 页建立符号连接：

```
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
```

默认情况下，Vim 的说明文档被安装在 `/usr/share/vim` 里。下面的这个符号链接使得可以通过 `/usr/share/doc/vim-8.1` 访问该文档，让它的位置与其它软件包的文档位置保持一致：

```
ln -sv ../vim/vim81/doc /usr/share/doc/vim-8.1
```

如果要把一个 X Window 系统安装在 LFS 系统上，可能得在安装完 X 系统后再重新编译 Vim。Vim 带有一个 GUI 版本，这个版本需要安装 X 和一些额外的库。想了解更多信息，请参考 Vim 文档和 BLFS <http://www.linuxfromscratch.org/blfs/view/8.4/postlfs/vim.html> 中 Vim 安装指导页。



## 6.71.2. 设置 Vim

默认情况下，vim 是以不兼容 vi 的模式运行的。这对于过去使用其它编辑器的用户可能是个新问题。下面列出了「非兼容性」设置以突出显示使用的新特性。它也提醒着那些想换成「兼容」模式的人，这是配置文件里第一个该被设置的地方。这非常有必要，因为它会改变其它的设置，而且覆写必须在这个设置之后。以下面的方式，创建一个默认的 vim 配置文件：

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

" Ensure defaults are set before customizing settings, not after
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
    set background=dark
endif

" End /etc/vimrc
EOF
```

设置 *set nocompatible* 让 vim 比 vi 兼容模式更有用。删掉「no」以保留旧的 vi 特性。设置 *set backspace=2* 让退格跨越换行、自动缩进和插入的开始。*syntax on* 参数使 vim 能高亮显示语法。设置 *set mouse* 让你能在 chroot 和远程连接的时候用鼠标粘帖文本。最后，带有 *set background=dark* 的 if 语句矫正了 vim 对于某些终端模拟器的背景颜色的估算。这让某些写在黑色背景上的程序的高亮色能有更好的调色方案。

用下面的命令可以获得其它选项的文档：

```
vim -c ':options'
```

### Note

默认情况下，Vim 只安装了英文的拼写检查文件。要想安装你想要的语言的拼写检查文件，请从 <ftp://ftp.vim.org/pub/vim/runtime/spell/> 下载你所用语言的 \*.spl 文件，可下可不下的 \*.sug 文件和文字编码。并把它们保存到 /usr/share/vim/vim81/spell/。

要使用这些文件，需要设置 /etc/vimrc 里的某些项，例如：

```
set spelllang=en,ru
set spell
```

想要了解更多信息，请阅读上方 URL 里对应 README 文件。

## 6.71.3. Vim 软件包内容

安装的软件: ex (链接到 vim), rview (链接到 vim), rvim (链接到 vim), vi (链接到 vim), view (链接到 vim), vim, vimdiff (链接到 vim), vimtutor, 和 xxd  
 安装的目录: /usr/share/vim

### 简要介绍

ex	以 ex 模式启动 vim
rview	是 view 的一个受限版本；不能启动 shell 命令，而且 view 无法暂停
rvim	是 vim 的一个受限版本；不能启动 shell 命令，而且 vim 无法暂停
vi	链接到 vim
view	以只读模式启动 vim
vim	就是这个编辑器
vimdiff	用 vim 编辑一个文件的两到三个版本并显示它们的区别
vimtutor	vim 基本热键和命令的教学

xxd 将给定文件进行十六进制转储，还可以还原；所以它可以被用于打二进制补丁

## 6.72. systemd-240

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

大致构建用时: 2.4 SBU  
所需磁盘空间: 226 MB

### 6.72.1. Installation of systemd

Apply a patch to fix two critical security vulnerabilities:

```
patch -Np1 -i ../systemd-240-security_fixes-2.patch
```

Create a symlink to work around missing xsltproc:

```
ln -sf /tools/bin/true /usr/bin/xsltproc
```

Because we have not yet installed the final version of Util-Linux, create links to the libraries in the appropriate location:

```
for file in /tools/lib/lib{blkid,mount,uuid}*; do
    ln -sf $file /usr/lib/
done
```

Set up the man pages:

```
tar -xf ../systemd-man-pages-240.tar.xz
```

Remove tests that cannot be built in chroot:

```
sed '177,$ d' -i src/resolve/meson.build
```

Remove an unneeded group, render, from the default udev rules:

```
sed -i 's/GROUP="render", //' rules/50-udev-default.rules.in
```

Prepare systemd for compilation:

```
mkdir -p build
cd      build

PKG_CONFIG_PATH="/usr/lib/pkgconfig:/tools/lib/pkgconfig" \
LANG=en_US.UTF-8 \
meson --prefix=/usr \
      --sysconfdir=/etc \
      --localstatedir=/var \
      -Dblkid=true \
      -Dbuildtype=release \
      -Ddefault-dnssec=no \
      -Dfirstboot=false \
      -Dinstall-tests=false \
      -Dkill-path=/bin/kill \
      -Dkmod-path=/bin/kmod \
      -Dldconfig=false \
      -Dmount-path=/bin/mount \
      -Drootprefix= \
      -Drootlibdir=/lib \
      -Dsplit-usr=true \
      -Dsulogin-path=/sbin/sulogin \
      -Dsysusers=false \
      -Dumount-path=/bin/umount \
      -Db_lto=false \
      ..
```

The meaning of the meson options:

*-D\*-path=\**

These switches provide location of binaries needed by systemd at runtime that have not yet been installed, or who's pkgconfig files are currently only in /tools/lib/pkgconfig.

*-Ddefault-dnssec=no*

This switch turns off the experimental DNSSEC support.

*-Dfirstboot=false*

This switch prevents installation of systemd services responsible for setting up the system for the first time. They are not useful for LFS because everything is done manually.

*-Dinstall-tests=false*

This switch prevents installation of the compiled tests.

*-Dldconfig=false*

This switch prevents installation of a systemd unit that runs ldconfig at boot, which is not useful for source distributions such as LFS and makes the boot time longer. Remove it if the described feature is desired.

*-Droot\**

These switches ensure that core programs and shared libraries are installed in the subdirectories of the root partition.

*-Dsplitt-usr=true*

This switch ensures that systemd will work on systems where /bin, /lib and /sbin directories are not symlinks to their /usr counterparts.

*-Dsysusers=false*

This switch prevents installation of systemd services responsible for setting up the /etc/group and /etc/passwd files. Both files were created earlier in this chapter.

Compile the package:

```
LANG=en_US.UTF-8 ninja
```

Install the package:

```
LANG=en_US.UTF-8 ninja install
```

Remove an unnecessary directory:

```
rm -rfv /usr/lib/rpm
rm -f /usr/bin/xsltproc
```

Create the /etc/machine-id file needed by systemd-journal:

```
systemd-machine-id-setup
```

Create the /lib/systemd/systemd-user-sessions script to allow unprivileged user logins without systemd-logind:

```
cat > /lib/systemd/systemd-user-sessions << "EOF"
#!/bin/bash
rm -f /run/nologin
EOF
chmod 755 /lib/systemd/systemd-user-sessions
```

## 6.72.2. Contents of systemd

Installed programs:	bootctl, busctl, coredumpctl, halt, hostnamectl, init, journalctl, kernel-install, localectl, loginctl, machinectl, networkctl, poweroff, reboot, runlevel, shutdown, systemctl, systemd-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-delta, systemd-detect-virt, systemd-escape, systemd-hwdb, systemd-inhibit, systemd-machine-id-setup, systemd-mount, systemd-notify, systemd-nspawn, systemd-path, systemd-resolve, systemd-run, systemd-socket-activate, systemd-stdio-bridge, systemd-tmpfiles, systemd-tty-ask-password-agent, telinit, timedatectl, and udevadm
Installed libraries:	libnss_myhostname.so.2, libnss_mymachines.so.2, libnss_resolve.so.2, libnss_systemd.so.2, libsystemd.so, libsystemd-shared-240.so, and libudev.so
Installed directories:	/etc/binfmt.d, /etc/init.d, /etc/kernel, /etc/modules-load.d, /etc/sysctl.d, /etc/systemd, /etc/tmpfiles.d, /etc/udev, /etc/xdg/systemd, /lib/systemd, /lib/udev, /usr/include/systemd, /usr/lib/binfmt.d, /usr/lib/kernel, /usr/lib/modules-load.d, /usr/lib/sysctl.d, /usr/lib/systemd, /usr/lib/tmpfiles.d, /usr/share/doc/systemd-240, /usr/share/factory, /usr/share/systemd, /var/lib/systemd, and /var/log/journal

## Short Descriptions

<code>bootctl</code>	Used to query the firmware and boot manager settings
<code>busctl</code>	Used to introspect and monitor the D-Bus bus
<code>coredumpctl</code>	Used to retrieve coredumps from the systemd journal
<code>halt</code>	Normally invokes shutdown with the <code>-h</code> option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file <code>/var/log/wtmp</code> that the system is being brought down
<code>hostnamectl</code>	Used to query and change the system hostname and related settings
<code>init</code>	The first process to be started when the kernel has initialized the hardware which takes over the boot process and starts all processes according to its configuration files
<code>journalctl</code>	Used to query the contents of the systemd journal
<code>kernel-install</code>	Used to add and remove kernel and initramfs images to and from <code>/boot</code>
<code>localectl</code>	Used to query and change the system locale and keyboard layout settings
<code>loginctl</code>	Used to introspect and control the state of the systemd Login Manager
<code>machinectl</code>	Used to introspect and control the state of the systemd Virtual Machine and Container Registration Manager
<code>networkctl</code>	Used to introspect the state of the network links as seen by <code>systemd-networkd</code>
<code>poweroff</code>	Tells the kernel to halt the system and switch off the computer (see <code>halt</code> )
<code>reboot</code>	Tells the kernel to reboot the system (see <code>halt</code> )
<code>runlevel</code>	Reports the previous and the current run-level, as noted in the last run-level record in <code>/var/run/utmp</code>
<code>shutdown</code>	Brings the system down in a secure way, signaling all processes and notifying all logged-in users
<code>systemctl</code>	Used to introspect and control the state of the systemd system and service manager
<code>systemd-analyze</code>	Used to determine system boot-up performance of the current boot
<code>systemd-ask-password</code>	Used to query a system password or passphrase from the user, using a question message specified on the command line
<code>systemd-cat</code>	Used to connect STDOUT and STDERR of a process with the Journal
<code>systemd-cgls</code>	Recursively shows the contents of the selected Linux control group hierarchy in a tree
<code>systemd-cgtop</code>	Shows the top control groups of the local Linux control group hierarchy, ordered by their CPU, memory and disk I/O load
<code>systemd-delta</code>	Used to identify and compare configuration files in <code>/etc</code> that override default counterparts in <code>/usr</code>
<code>systemd-detect-virt</code>	Detects execution in a virtualized environment
<code>systemd-escape</code>	Used to escape strings for inclusion in systemd unit names
<code>systemd-hwdb</code>	Used to manage hardware database (hwdb)
<code>systemd-inhibit</code>	Used to execute a program with a shutdown, sleep or idle inhibitor lock taken
<code>systemd-machine-id-setup</code>	Used by system installer tools to initialize the machine ID stored in <code>/etc/machine-id</code> at install time with a randomly generated ID

<code>systemd-mount</code>	A tool to temporarily mount or auto-mount a drive.
<code>systemd-notify</code>	Used by daemon scripts to notify the init system about status changes
<code>systemd-nspawn</code>	Used to run a command or OS in a light-weight namespace container
<code>systemd-path</code>	Used to query system and user paths
<code>systemd-resolve</code>	Used to resolve domain names, IPV4 and IPV6 addresses, DNS resource records, and services
<code>systemd-run</code>	Used to create and start a transient <code>.service</code> or a <code>.scope</code> unit and run the specified command in it
<code>systemd-socket-activate</code>	A tool to listen on socket devices and launch a process upon connection.
<code>systemd-tmpfiles</code>	Creates, deletes and cleans up volatile and temporary files and directories, based on the configuration file format and location specified in <code>tmpfiles.d</code> directories
<code>systemd-tty-ask-password-agent</code>	Used to list or process pending systemd password requests
<code>telinit</code>	Tells init which run-level to change to
<code>timedatectl</code>	Used to query and change the system clock and its settings
<code>udevadm</code>	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
<code>libsystemd</code>	The main systemd utility library
<code>libudev</code>	A library to access udev device information

## 6.73. D-Bus-1.12.12

D-Bus is a message bus system, a simple way for applications to talk to one another. D-Bus supplies both a system daemon (for events such as "new hardware device added" or "printer queue changed") and a per-user-login-session daemon (for general IPC needs among user applications). Also, the message bus is built on top of a general one-to-one message passing framework, which can be used by any two applications to communicate directly (without going through the message bus daemon).

大致构建用时: 0.2 SBU  
所需磁盘空间: 18 MB

### 6.73.1. Installation of D-Bus

Prepare D-Bus for compilation:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --localstatedir=/var \
            --disable-static \
            --disable-doxygen-docs \
            --disable-xml-docs \
            --docdir=/usr/share/doc/dbus-1.12.12 \
            --with-console-auth-dir=/run/console
```

The meaning of the configure options:

`--with-console-auth-dir=/run/console`

This specifies the location of the ConsoleKit auth directory.

Compile the package:

```
make
```

This package does come with a test suite, but it requires several packages that are not included in LFS. Instructions for running the test suite can be found in the BLFS book at <http://www.linuxfromscratch.org/blfs/view/8.4/general/dbus.html>.

Install the package:

```
make install
```

The shared library needs to be moved to `/lib`, and as a result the `.so` file in `/usr/lib` will need to be recreated:

```
mv -v /usr/lib/libdbus-1.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libdbus-1.so) /usr/lib/libdbus-1.so
```

Create a symlink, so that D-Bus and systemd can use the same `machine-id` file:

```
ln -sfv /etc/machine-id /var/lib/dbus
```

### 6.73.2. Contents of D-Bus

Installed programs:	dbus-cleanup-sockets, dbus-daemon, dbus-launch, dbus-monitor, dbus-run-session, dbus-send, dbus-test-tool, dbus-update-activation-environment, and dbus-uuidgen
Installed libraries:	libdbus-1.{a,so}
Installed directories:	/etc/dbus-1, /usr/include/dbus-1.0, /usr/lib/dbus-1.0, /usr/share/dbus-1, /usr/share/doc/dbus-1.12.12, and /var/lib/dbus

### Short Descriptions

dbus-cleanup-sockets	Used to clean up leftover sockets in a directory
dbus-daemon	The D-Bus message bus daemon
dbus-launch	Starts dbus-daemon from a shell script
dbus-monitor	Monitors messages passing through a D-Bus message bus
dbus-run-session	Starts a session bus instance of dbus-daemon from a shell script and starts a specified program in that session

dbus-send	Sends a message to a D-Bus message bus
dbus-test-tool	A tool to help packages test D-Bus
dbus-update-activation-environment	Updates environment variables that will be set for D-Bus session services
dbus-uuidgen	Generates a universally unique ID
libdbus-1	Contains API functions used to communicate with the D-Bus message bus



## 6.74. Procps-ng-3.3.15

Procps-ng 软件包包含监视进程的程序。

大致构建用时: 0.1 SBU  
所需磁盘空间: 17 MB

### 6.74.1. 安装 Procps-ng

准备编译 procps-ng :

```
./configure --prefix=/usr \
            --exec-prefix= \
            --libdir=/usr/lib \
            --docdir=/usr/share/doc/procps-ng-3.3.15 \
            --disable-static \
            --disable-kill \
            --with-systemd
```

配置选项的含义：

`--disable-kill`

该选项将不会编译已经由 Util-linux 软件包安装了的 kill 命令。

编译软件包：

```
make
```

对于 LFS，测试套件需要自定义某些更改。移除当脚本不使用 tty 设备时失败的测试，同时修复另两个失败。用下面的命令运行测试套件：

```
sed -i -r 's|(pmap_initname)\\$|\\1|' testsuite/pmap.test/pmap.exp
sed -i '/set tty/d' testsuite/pkill.test/pkill.exp
rm testsuite/pgrep.test/pgrep.exp
make check
```

安装软件包：

```
make install
```

最后，如果 /usr 没有挂载的话，移动重要文件到一个可以找到的位置。

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) /usr/lib/libprocps.so
```

### 6.74.2. Procps-ng 软件包内容

安装的程序: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, 和 watch  
安装的库: libprocps.so  
安装目录: /usr/include/proc 和 /usr/share/doc/procps-ng-3.3.15

#### 简要介绍

free	报告系统中空闲和使用的内存容量（包括物理和交换内存）
pgrep	根据名称和其它属性查找进程
pidof	报告指定程序的 PID
pkill	根据名称和其它属性给进程发送信号
pmap	报告指定进程的内存映射情况
ps	列出正在运行的进程
pwdx	报告进程的当前工作目录
slabtop	实时显示内核 slab 缓存信息
sysctl	运行时修改内核参数
tload	打印当前系统平均负荷曲线图
top	显示最 CPU 密集型进程列表；它可以实时地连续查看处理器活动

uptime	报告系统运行时长、登录用户数目以及系统平均负荷
vmstat	报告虚拟内存统计信息、给出关于进程、内存、分页、块输入/输出(IO)、陷阱以及 CPU 活动的信息
w	显示当前登录的用户、以及登录地点和时间
watch	重复运行指定命令，显示输出的第一个整屏；这允许用户查看随着时间的输出变化
libprocps	包含该软件包大部分程序使用的函数

## 6.75. Util-linux-2.33.1

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

大致构建用时: 1.5 SBU  
所需磁盘空间: 214 MB

### 6.75.1. 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. First create a directory to enable storage for the hwclock program:

```
mkdir -pv /var/lib/hwclock
```

### 6.75.2. Installation of Util-linux

Remove the earlier created symlinks:

```
rm -vf /usr/include/{blkid,libmount,uuid}
```

Prepare Util-linux for compilation:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime \
    --docdir=/usr/share/doc/util-linux-2.33.1 \
    --disable-chfn-chsh \
    --disable-login \
    --disable-nologin \
    --disable-su \
    --disable-setpriv \
    --disable-runuser \
    --disable-pylibmount \
    --disable-static \
    --without-python
```

The `--disable` and `--without` options prevent warnings about building components that require packages not in LFS or are inconsistent with programs installed by other packages.

Compile the package:

```
make
```

If desired, run the test suite as a non-root user:

#### Warning

Running the test suite as the root user can be harmful to your system. To run it, the `CONFIG_SCSI_DEBUG` option for the kernel must be available in the currently running system, and must be built as a module. Building it into the kernel will prevent booting. For complete coverage, other BLFS packages must be installed. If desired, this test can be run after rebooting into the completed LFS system and running:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

Install the package:

```
make install
```

## 6.75.3. Contents of Util-linux

Installed programs:	addpart, agetty, blkdiscard, blkid, blockdev, cal, cfdisk, chcpu, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipc, isosize, kill, last, lastb (link to last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lslocks, lslogins, mcookie, msg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swapon, swaplabel, swapon (link to swapon), swapon, switch_root, tailf, taskset, ul, umount, uname26, unshare, utmpdump, uuid, uuidgen, wall, wdcctl, whereis, wipefs, x86_64, and zramctl
Installed libraries:	libblkid.so, libfdisk.so, libmount.so, libsmartcols.so, and libuuid.so
Installed directories:	/usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.33.1, and /var/lib/hwclock

## 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
blkdiscard	Discards sectors on a device
blkid	A command line utility to locate and print block device attributes
blockdev	Allows users to call block device ioctls from the command line
cal	Displays a simple calendar
cfdisk	Manipulates the partition table of the given device
chcpu	Modifies the state of CPUs
choom	Displays and adjusts OOM-killer score
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
delpart	Asks the Linux kernel to remove a partition
dmesg	Dumps the kernel boot messages
eject	Ejects removable media
fallocate	Preallocates space to a file
fdformat	Low-level formats a floppy disk
fdisk	Manipulates the partition table of the given device
findfs	Finds a file system by label or Universally Unique Identifier (UUID)
findmnt	Is a command line interface to the libmount library for work with mountinfo, fstab and mtab files
flock	Acquires a file lock and then executes a command with the lock held
fsck	Is used to check, and optionally repair, file systems
fsck.cramfs	Performs a consistency check on the Cramfs file system on the given device
fsck.minix	Performs a consistency check on the Minix file system on the given device
fsfreeze	Is a very simple wrapper around FIFREEZE/FITHAW ioctl kernel driver operations
fstrim	Discards unused blocks on a mounted filesystem
getopt	Parses options in the given command line

hexdump	Dumps the given file in hexadecimal or in another given format
hwclock	Reads or sets the system's hardware clock, also called the Real-Time Clock (RTC) or Basic Input-Output System (BIOS) clock
i386	A symbolic link to setarch
ionice	Gets or sets the io scheduling class and priority for a program
ipcmk	Creates various IPC resources
ipcrm	Removes the given Inter-Process Communication (IPC) resource
ipcs	Provides IPC status information
isozsize	Reports the size of an iso9660 file system
kill	Sends signals to processes
last	Shows which users last logged in (and out), searching back through the <code>/var/log/wtmp</code> file; it also shows system boots, shutdowns, and run-level changes
lastb	Shows the failed login attempts, as logged in <code>/var/log/btmp</code>
ldattach	Attaches a line discipline to a serial line
linux32	A symbolic link to setarch
linux64	A symbolic link to setarch
logger	Enters the given message into the system log
look	Displays lines that begin with the given string
losetup	Sets up and controls loop devices
lsblk	Lists information about all or selected block devices in a tree-like format
lscpu	Prints CPU architecture information
lsipc	Prints information on IPC facilities currently employed in the system
lslocks	Lists local system locks
lslogins	Lists information about users, groups and system accounts
mcookie	Generates magic cookies (128-bit random hexadecimal numbers) for xauth
mesg	Controls whether other users can send messages to the current user's terminal
mkfs	Builds a file system on a device (usually a hard disk partition)
mkfs.bfs	Creates a Santa Cruz Operations (SCO) bfs file system
mkfs.cramfs	Creates a cramfs file system
mkfs.minix	Creates a Minix file system
mkswap	Initializes the given device or file to be used as a swap area
more	A filter for paging through text one screen at a time
mount	Attaches the file system on the given device to a specified directory in the file-system tree
mountpoint	Checks if the directory is a mountpoint
namei	Shows the symbolic links in the given pathnames
nsenter	Runs a program with namespaces of other processes
partx	Tells the kernel about the presence and numbering of on-disk partitions
pg	Displays a text file one screen full at a time
pivot_root	Makes the given file system the new root file system of the current process
prlimit	Get and set a process' resource limits
raw	Bind a Linux raw character device to a block device
readprofile	Reads kernel profiling information
rename	Renames the given files, replacing a given string with another
renice	Alters the priority of running processes
resizepart	Asks the Linux kernel to resize a partition
rev	Reverses the lines of a given file
rtcwake	Used to enter a system sleep state until specified wakeup time

<code>script</code>	Makes a typescript of a terminal session
<code>scriptreplay</code>	Plays back typescripts using timing information
<code>setarch</code>	Changes reported architecture in a new program environment and sets personality flags
<code>setsid</code>	Runs the given program in a new session
<code>setterm</code>	Sets terminal attributes
<code>sfdisk</code>	A disk partition table manipulator
<code>sulogin</code>	Allows <code>root</code> to log in; it is normally invoked by <code>init</code> when the system goes into single user mode
<code>swapon</code>	Enables devices and files for paging and swapping and lists the devices and files currently in use
<code>swapoff</code>	Disables devices and files for paging and swapping
<code>swapon</code>	Enables devices and files for paging and swapping and lists the devices and files currently in use
<code>switch_root</code>	Switches to another filesystem as the root of the mount tree
<code>tailf</code>	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
<code>taskset</code>	Retrieves or sets a process' CPU affinity
<code>ul</code>	A filter for translating underscores into escape sequences indicating underlining for the terminal in use
<code>umount</code>	Disconnects a file system from the system's file tree
<code>uname26</code>	A symbolic link to <code>setarch</code>
<code>unshare</code>	Runs a program with some namespaces unshared from parent
<code>utmpdump</code>	Displays the content of the given login file in a more user-friendly format
<code>uuid</code>	A daemon used by the UUID library to generate time-based UUIDs in a secure and guaranteed-unique fashion
<code>uuidgen</code>	Creates new UUIDs. Each new UUID can reasonably be considered unique among all UUIDs created, on the local system and on other systems, in the past and in the future
<code>wall</code>	Displays the contents of a file or, by default, its standard input, on the terminals of all currently logged in users
<code>wdctl</code>	Shows hardware watchdog status
<code>whereis</code>	Reports the location of the binary, source, and man page for the given command
<code>wipefs</code>	Wipes a filesystem signature from a device
<code>x86_64</code>	A symbolic link to <code>setarch</code>
<code>zramctl</code>	A program to set up and control <code>zram</code> (compressed ram disk) devices
<code>libblkid</code>	Contains routines for device identification and token extraction
<code>libfdisk</code>	Contains routines for manipulating partition tables
<code>libmount</code>	Contains routines for block device mounting and unmounting
<code>libsmartcols</code>	Contains routines for aiding screen output in tabular form
<code>libuuid</code>	Contains routines for generating unique identifiers for objects that may be accessible beyond the local system

## 6.76. E2fsprogs-1.44.5

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

大致构建用时: 1.6 SBU  
所需磁盘空间: 96 MB

### 6.76.1. Installation of E2fsprogs

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

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

Prepare E2fsprogs for compilation:

```
../configure --prefix=/usr      \
              --bindir=/bin     \
              --with-root-prefix="" \
              --enable-elf-shlibs \
              --disable-libblkid \
              --disable-libuuid  \
              --disable-uuid     \
              --disable-fsck
```

The meaning of the environment variable and configure options:

`--with-root-prefix=""` and `--bindir=/bin`

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

`--enable-elf-shlibs`

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

`--disable-*`

This prevents E2fsprogs from building and installing the `libuuid` and `libblkid` libraries, the `uuid` daemon, and the `fsck` wrapper, as Util-Linux installs more recent versions.

Compile the package:

```
make
```

To run the tests, 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, be sure to enable sufficient swap space for the test. See Section#2.5, “在分区上创建文件系统” and Section#2.7, “挂载新分区” 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/{libcom_err,libe2p,libext2fs,libss}.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
```

## 6.76.2. Contents of E2fsprogs

Installed programs: badblocks, chattr, compile\_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk\_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, mklost+found, resize2fs, and tune2fs

Installed libraries: libcom\_err.so, libe2p.so, libext2fs.so, and libss.so

Installed directories: /usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/share/et, and /usr/share/ss

## Short Descriptions

badblocks	Searches a device (usually a disk partition) for bad blocks
chattr	Changes the attributes of files on an <code>ext2</code> file system; it also changes <code>ext3</code> file systems, the journaling version of <code>ext2</code> file systems
compile_et	An error table compiler; it converts a table of error-code names and messages into a C source file suitable for use with the <code>com_err</code> library
debugfs	A file system debugger; it can be used to examine and change the state of an <code>ext2</code> file system
dumpe2fs	Prints the super block and blocks group information for the file system present on a given device
e2freefrag	Reports free space fragmentation information
e2fsck	Is used to check, and optionally repair <code>ext2</code> file systems and <code>ext3</code> file systems
e2image	Is used to save critical <code>ext2</code> file system data to a file
e2label	Displays or changes the file system label on the <code>ext2</code> file system present on a given device
e2undo	Replays the undo log <code>undo_log</code> for an <code>ext2</code> / <code>ext3</code> / <code>ext4</code> filesystem found on a device [This can be used to undo a failed operation by an <code>e2fsprogs</code> program.]
e4defrag	Online defragmenter for <code>ext4</code> filesystems
filefrag	Reports on how badly fragmented a particular file might be
fsck.ext2	By default checks <code>ext2</code> file systems and is a hard link to <code>e2fsck</code>
fsck.ext3	By default checks <code>ext3</code> file systems and is a hard link to <code>e2fsck</code>
fsck.ext4	By default checks <code>ext4</code> file systems and is a hard link to <code>e2fsck</code>
fsck.ext4dev	By default checks <code>ext4</code> development file systems and is a hard link to <code>e2fsck</code>
logsave	Saves the output of a command in a log file
lsattr	Lists the attributes of files on a second extended file system
mk_cmds	Converts a table of command names and help messages into a C source file suitable for use with the <code>libss</code> subsystem library
mke2fs	Creates an <code>ext2</code> or <code>ext3</code> file system on the given device
mkfs.ext2	By default creates <code>ext2</code> file systems and is a hard link to <code>mke2fs</code>
mkfs.ext3	By default creates <code>ext3</code> file systems and is a hard link to <code>mke2fs</code>
mkfs.ext4	By default creates <code>ext4</code> file systems and is a hard link to <code>mke2fs</code>
mkfs.ext4dev	By default creates <code>ext4</code> development file systems and is a hard link to <code>mke2fs</code>
mklost+found	Used to create a <code>lost+found</code> directory on an <code>ext2</code> file system; it pre-allocates disk blocks to this directory to lighten the task of <code>e2fsck</code>
resize2fs	Can be used to enlarge or shrink an <code>ext2</code> file system
tune2fs	Adjusts tunable file system parameters on an <code>ext2</code> file system
libcom_err	The common error display routine



<code>libe2p</code>	Used by <code>dumpe2fs</code> , <code>chattr</code> , and <code>lsattr</code>
<code>libext2fs</code>	Contains routines to enable user-level programs to manipulate an <code>ext2</code> file system
<code>libss</code>	Used by <code>debugfs</code>

## 6.77. 关于调试符号

默认情况下大多数程序和库的编译带有调试符号。(类似 gcc 的 `-g` 选项。)这意味着当你调试一个包含调试信息的已编译的程序或库时,调试程序不仅能提供内存地址,还能提供变量和实例的名字。

然而,包含这些调试符号明显的增大了程序或库。下面这个例子说明了这些符号有多么占地方:

- 有调试符号的二进制 bash : 1200 KB
- 无调试符号的二进制 bash : 480 KB
- 有调试符号的 Glibc 和 GCC 文件 (`/lib` 和 `/usr/lib`) : 87 MB
- 无调试符号的 Glibc 和 GCC 文件 : 16 MB

大小可能会因为所使用的编译器和 C 语言库的不同而改变,但是当比较有无调试符号的程序时,大小可能相差 2 到 5 倍。

因为大多数用户从来不会在他们的系统软件上使用调试器,没了这些调试符号可以省下很多磁盘空间。下一页将会告诉你如何剥离程序和库中所有的调试符号。

## 6.78. 再次清理无用内容

这个部分是可选的。如果预期的用户不是一个程序员或者不打算对系统软件进行任何调试,通过从二进制文件和库中删除调试符号能减少 90MB 的系统大小。除了不能完全调试软件,这不会导致任何不便。

大部分人使用下面提到的命令并不会感到任何困难。然而,很容易出现错误并导致新的系统不可用,因此在运行 `strip` 命令之前,对当前状态的 LFS 系统进行备份是个好主意。

首先将选定库的调试符号文件分开放置。如果要在后续的 BLFS 中用 `valgrind` 或 `gdb` 做回归测试,那么调试信息还有用武之地。

```
save_lib="ld-2.29.so libc-2.29.so libpthread-2.29.so libthread_db-1.0.so"

cd /lib

for LIB in $save_lib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

save_usrlib="libquadmath.so.0.0.0 libstdc++.so.6.0.25
            libitm.so.1.0.0 libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    strip --strip-unneeded $LIB
    objcopy --add-gnu-debuglink=$LIB.dbg $LIB
done

unset LIB save_lib save_usrlib
```

在进行清理无用内容之前,格外注意确保要删除的二进制文件没有正在运行:

```
exec /tools/bin/bash
```

现在可以安心的清除二进制文件和库:

```
/tools/bin/find /usr/lib -type f -name \*.a \
    -exec /tools/bin/strip --strip-debug {} ';'

/tools/bin/find /lib /usr/lib -type f \( -name \*.so* -a ! -name \*.dbg \) \
    -exec /tools/bin/strip --strip-unneeded {} ';'

/tools/bin/find /{bin,sbin} /usr/{bin,sbin,libexec} -type f \
    -exec /tools/bin/strip --strip-all {} ';'


```

该命令会报告有很大数目的文件不能识别它们的格式。你可以安全地忽略这些警告。这些警告表示这些文件是脚本而不是二进制文件。

## 6.79. 清理

最后，清除运行测试留下来的多余文件：

```
rm -rf /tmp/*
```

现在，登出后用以下新的 chroot 命令重新进入 chroot 环境。在此以后当需要进入 chroot 环境时，都是用这个新的 chroot 命令：

```
logout

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

这样做的原因是不再需要 /tools 中的程序。因此你可以删除 /tools 目录。

### Note

移除 /tools 也会删除用于运行工具链测试的 Tcl、Expect 和 DejaGNU 的临时复制。如果你在后面还需要这些程序，需要重新编译并安装它们。BLFS 手册有关于这些的指令（请查看<http://www.linuxfromscratch.org/blfs/>）。

如果通过手动或者重启卸载了虚拟内核文件系统，重新进入 chroot 的时候确保挂载了虚拟内核文件系统。在 Section#6.2.2, “挂载和激活 /dev”和 Section#6.2.3, “挂载虚拟文件系统”中介绍了该过程。

还有一些此章之前为了一些软件包的回归测试而留下的静态库。这些库来自 binutils、bzip2、e2fsprogs、flex、libtool 和 zlib。如果想删的话，现在就删：

```
rm -f /usr/lib/lib{bfd,opcodes}.a
rm -f /usr/lib/libbz2.a
rm -f /usr/lib/lib{com_err,e2p,ext2fs,ss}.a
rm -f /usr/lib/libltdl.a
rm -f /usr/lib/libfl.a
rm -f /usr/lib/libz.a
```

还有几个安装在 /usr/lib 和 /usr/libexec 目录下的文件，文件的扩展名为 .la。这些是「libtool 归档」文件，在 Linux 系统中通常不需要它们。这些都是没有必要的东西。想要删除的话，运行：

```
find /usr/lib /usr/libexec -name *.la -delete
```

关于 libtool 归档文件的更多信息，参考 BLFS 段落「关于 Libtool 归档 (.la) 文件」。

## Chapter 7. 基本系统配置

### 7.1. 简介

这一章将会讲解配置文件和 systemd 服务。首先，给出了设置网络需要的通用配置文件。

- Section#7.2, “通用网络配置”
- Section#7.2.3, “配置系统主机名称”
- Section#7.2.4, “自定义 /etc/hosts 文件”

然后，讨论可能影响正确的设备配置的问题。

- Section#7.3, “设备与模块管理概述”
- Section#7.4, “设备管理”

第三步，配置系统时间和键盘布局。

- Section#7.5, “配置系统时间”
- Section#7.6, “配置 Linux 控制台”

第四步，简单介绍当用户登录系统时用到的脚本和配置文件。

- Section#7.7, “配置系统语言环境”
- Section#7.8, “Creating the /etc/inputrc File”

最后，配置 systemd。

- Section#7.10, “systemd Usage and Configuration”

### 7.2. 通用网络配置

本节仅在需要配置网卡时参考。

#### 7.2.1. 网络接口配置文件

自版本 209 开始，systemd 提供了一个名为 systemd-networkd 的命令用于处理基本的网络配置。另外，自版本 213 开始，DNS 名称解析可用 systemd-resolved 代替静态的 /etc/resolv.conf 文件来解决。默认情况，两种服务都将被启用。

systemd-networkd (和 systemd-resolved 的配置文件可能在 /usr/lib/systemd/network 或 /etc/systemd/network 中。/etc/systemd/network 中文件比 /usr/lib/systemd/network 中的有更高的优先级。配置文件类型有三种：.link, .netdev 和 .network 文件。可以通过查阅 man 手册的 systemd-link(5), systemd-netdev(5) 和 systemd-network(5) 获取更多关于这些配置文件的详细介绍。

##### 7.2.1.1. 网络设备命名

udev 通常会根据系统物理特性分配接口名称。例如，enp2s1。如果你不确定你的接口名称，你可以在系统启动后运行 ip link 查看。

对于大多数系统，每种连接只会有一种网络接口。例如，传统有线连接的接口名 eth0。而无线连接的名称通常是 wifi0 或 wlan0。

如果你更青睐传统或是自定义的网络接口名称，有三种实现的方法：

- 为默认的策略隐蔽 udev 的 .link 文件：

```
ln -s /dev/null /etc/systemd/network/99-default.link
```

- 创建手动命名规则，比方说，将接口命名成「internet0」，「dmz0」或「lan0」这样。为此，请在 /etc/systemd/network/ 中创建 .link 文件，为其中的一个，一些，或者说你全部的接口赋予明确的名字或是更妥善的命名规则。示例：

```
cat > /etc/systemd/network/10-ether0.link << "EOF"
[Match]
# Change the MAC address as appropriate for your network device
MACAddress=12:34:45:78:90:AB

[Link]
Name=ether0
EOF
```

参考 man 手册 `systemd.link(5)` 获取更多信息。

- 在 `/boot/grub/grub.cfg` 中，给内核命令行传递 `net.ifnames=0` 选项。

### 7.2.1.2. 静态 IP 的配置

以下为设置静态 IP 而创建的基础配置文件（同时用到了 `systemd-networkd` 和 `systemd-resolved`）：

```
cat > /etc/systemd/network/10-eth-static.network << "EOF"
[Match]
Name=<network-device-name>

[Network]
Address=192.168.0.2/24
Gateway=192.168.0.1
DNS=192.168.0.1
Domains=<Your Domain Name>
EOF
```

如果你拥有的 DNS 服务器超过了一个，可以为其添加多个 DNS 条目。然而，如果你打算用静态的 `/etc/resolv.conf` 文件，就不要添加 DNS 或域名条目。

### 7.2.1.3. DHCP 的配置

通过以下命令，你可以创建用于设置 IPv4 DHCP 的基础配置文件：

```
cat > /etc/systemd/network/10-eth-dhcp.network << "EOF"
[Match]
Name=<network-device-name>

[Network]
DHCP=ipv4

[DHCP]
UseDomains=true
EOF
```

## 7.2.2. 创建 `/etc/resolv.conf` 文件

如果你的系统需要连接到互联网，它需要利用 DNS 服务将互联网域名解析为实际的 IP 地址，反之亦然。最好的方法是将从 ISP 或者是网络管理员那里取得的 DNS 服务器地址填入 `/etc/resolv.conf`。

### 7.2.2.1. `systemd-resolved` 的配置

#### Note

如果你使用其他方式来配置你的网络接口（例如：`ppp`，`network-manager`，等），或是任何类型的本地解析器（例如：`bind`，`dnsmasq`，等），或是任何生成 `/etc/resolv.conf` 文件的软件（例如：`resolvconf`），也就别用 `systemd-resolved` 服务了。

用 `systemd-resolved` 配置 DNS 时，会创建 `/run/systemd/resolve/resolv.conf` 文件。并在 `/etc` 中创建一个指向生成文件的软链接：

```
ln -sfv /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

### 7.2.2.2. 静态 `resolv.conf` 的配置

如果需要静态的 `/etc/resolv.conf` 文件，请使用以下命令：

```
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
```

domain 声明可以忽略或者以 search 声明替换。参考 man 手册的 resolv.conf 部分获得更多信息。

其中, `<IP address of the nameserver>` 替换为最合适的 DNS 的 IP 地址。通常会有多个条目(需要备选服务器具有相关兼容性)。如果你只需要一台 DNS 服务器, 请不要输入第二行 nameserver 的内容。该 IP 地址也可以是本地网络中的一台路由。

### Note

Google 公开的 DNS 解析服务器地址 IPv4 的为: `8.8.8.8` 和 `8.8.4.4`。IPv6 的为 `2001:4860:4860::8888` 和 `2001:4860:4860::8844`。

114 DNS: `114.114.114.114` 和 `114.114.115.115`。

阿里 DNS: `223.5.5.5` 和 `223.6.6.6`。

百度 DNS: `180.76.76.76`

OpenDNS: `208.67.220.220`

## 7.2.3. 配置系统主机名称

在系统启动过程中, `/etc/hostname` 文件用于创建系统的主机名称。

通过以下命令创建 `/etc/hostname` 文件:

```
echo "<lfs>" > /etc/hostname
```

`<lfs>` 替换为你想要设置的名称。请不要输入完整域名 (Fully Qualified Domain Name, FQDN), 那应该是放在 `/etc/hosts` 文件中的信息。

## 7.2.4. 自定义 `/etc/hosts` 文件

决定完整域名 (Fully-Qualified Domain Name, FQDN), 和可在文件 `/etc/hosts` 中使用的别名。如果使用的是静态地址, 你还需要决定 IP 地址。hosts 文件中条目的语法为:

```
IP_address myhost.example.org aliases
```

除非电脑在互联网中可见 (例如, 拥有注册的域名且分配有有效的 IP 地址——大多数用户并没有这些), 请确保 IP 地址位于有效的私有网络 IP 地址段。有效区间是:

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 可以是 16-31 之间的任何数字。y 可以是 0-255 之间的任何数字。

有效的私有 IP 可以是 192.168.1.1。而与之相配完整域名可以是 `lfs.example.org`。

即使没有网卡, 一个有效的完整仍然有其必要。它的存在可以确保程序正常运行。

如果使用的是 DHCP, DHCPv6, IPv6 自动配置, 或者说不配置网卡的话, 通过以下命令可以创建 `/etc/hosts` 文件:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost
127.0.1.1 <FQDN> <HOSTNAME>
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# End /etc/hosts
EOF
```

`::1` 相对与 IPv4 的 127.0.0.1, 是 IPv6 的回环地址。127.0.1.1 专门为 FQDN 保留的回环地址。

如果使用的是静态地址，可代替的使用以下命令创建 `/etc/hosts` 文件。

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.0.2> <FQDN> <HOSTNAME> [alias1] [alias2] ...
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# End /etc/hosts
EOF
```

其中 `<192.168.0.2>`，`<FQDN>`，和 `<HOSTNAME>` 的值需要根据具体的用途或需求更改（如果网络/系统管理委员会分配了 IP 地址，且机器将会接入现有的网络）。那可选的别名可以忽略。

## 7.3. 设备与模块管理概述

在 Chapter#6 我们构建 `systemd` 时，已经安装了 `udev` 包。在我们详细说明 `udev` 系统的用法之前，我们先大致了解早先的设备控制方式。

传统的 Linux 不管硬件是否真实存在，都以创建静态设备的方法来处理硬件，因此需要在 `/dev` 目录下创建大量的设备节点文件（有时会有上千个）。这通常由 `MAKEDEV` 脚本完成，它通过大量调用 `mknod` 程序为这个世界上可能存在的每一个设备建立对应的主设备号和次设备号。

而使用 `udev` 方法，只有当内核检测到硬件接入，才会建立对应的节点文件。因为需要在系统启动的时候重新建立设备节点文件，所以将它存储在 `devtmpfs` 文件系统中（完全存在于内存中的虚拟文件系统）。设备节点文件无需太多的空间，所以占用的内存也很小。

### 7.3.1. 历史

2000 年 2 月，一种名叫 `devfs` 的文件系统被合并到 2.3.46 版本的内核之中，而在 2.4 系列的稳定内核中基本可用。尽管它存在于内核代码中，但是这种动态创建设备的方法却从来都没有到核心开发者的大力支持。

问题存在于它处理设备的检测、创建和命令的方式。其中最大的问题莫过于它对设备节点的命名方式。大部分开发者的观点是：设备的命名应该由系统的所有者决定，而不是开发者。而且 `devfs` 存在严重的竞争条件 (race condition) 问题，如不对内核做大量的修改就无法修正这一问题。最终，因为缺乏有效的维护，在 2006 年 6 月终被移出内核源代码。

后来，有一种新的虚拟文件系统 `sysfs` 在 2.5 系列测试版本内核中引入，并且加入了 2.6 系列的稳定版本内核之中。`sysfs` 系统的任务就是将系统中的硬件配置状态导出至用户空间，而这给开发一种运行于用户空间的新型 `devfs` 系统带来了可能。

### 7.3.2. udev 实现

#### 7.3.2.1. Sysfs

上文简单的提及了 `sysfs` 文件系统。有些人可能会问，`sysfs` 到底是如何知道当前系统有哪些设备、这些设备又该使用什么设备号呢。对于那些已经编译进内核的设备，会在内核检测到时被直接注册为 `sysfs` 对象（由 `devtmpfs` 内建）。对于编译为内核模块的设备，将会在模块载入的时候注册。一旦 `sysfs` 文件系统挂载到 `/sys`，已经在 `sysfs` 注册的硬件数据就可以被用户空间的进程使用，随后也就可以被 `udev` 处理了（包括对设备节点进行修改）。

#### 7.3.2.2. 设备节点的创建

设备文件是通过内核中的 `devtmpfs` 文件系统创建的。任何想要注册的设备都需要通过 `devtmpfs`（经由驱动程序核心）实现。每当一个 `devtmpfs` 实例挂载到 `/dev`，就会建立一个设备节点文件，它拥有固定的名称、权限以及所有者。

很短的时间之后，内核将给 `udev` 一个 `uevent`。基于 `/etc/udev/rules.d`、`/lib/udev/rules.d` 和 `/run/udev/rules.d` 目录内文件指定的规则，`udev` 将会建立到设备节点文件的额外符号链接，这有可能更改其权限、所有者和所在组，或者是更改 `udev` 内建接口（名称）。

这三个文件夹中的规则文件都应以数字编号，并会被一起处理。当发现一个新的设备时，若 `udev` 无法找到对应的规则，将会使用 `devtmpfs` 中初始的权限以及所有者。

### 7.3.2.3. 加载模块

编译成模块的设备驱动可能会包含别名。别名可以通过 `modinfo` 命令查看，一般是模块支持的特定总线的设备描述符。举个例子，驱动 `snd-fm801` 支持厂商 ID 为 `0x1319` 以及设备 ID 为 `0x0801` 的设备，它包含一个「`pci:v00001319d00000801sv*sd*bc04sc01i*`」的别名，总线驱动导出该驱动别名并通过 `sysfs` 处理相关设备。例如，文件 `/sys/bus/pci/devices/0000:00:0d.0/modalias` 应该会包含字符串「`pci:v00001319d00000801sv00001319sd00001319bc04sc01i00`」。udev 采用的默认规则会让 `udev` 根据 `uevent` 环境变量 `MODALIAS` 的内容（它应该和 `sysfs` 里的 `modalias` 文件内容一样）调用 `/sbin/modprobe`，这样就可以加载在通配符扩展后能和这个字符串一致的所有模块。

这个例子意味着，除了 `snd-fm801` 之外，一个已经废弃的（不是我们所希望的）驱动 `forte` 如果存在的话也会被加载。下面有几种可以避免加载多余驱动的方式。

内核本身也能够根据需要加载网络协议，文件系统以及 NLS 支持模块。

### 7.3.2.4. 处理热插拔/动态设备

在你插入一个设备时，例如一个通用串行总线（USB）MP3 播放器，内核检测到设备已连接就会生成一个 `uevent`。这个 `uevent` 随后会被上面所说的 `udev` 处理。

## 7.3.3. 加载模块和创建设备时可能碰到的问题

在自动创建设备节点时可能会碰到一些问题。

### 7.3.3.1. 内核模块没有自动加载

`udev` 只会加载包含有特定总线别名而且已经被总线驱动导出到 `sysfs` 下的模块。在其它情况下，你应该考虑用其它方式加载模块。采用 Linux-4.20.12，`udev` 可以加载编写合适的 `INPUT`、`IDE`、`PCI`、`USB`、`SCSI`、`SERIO` 和 `FireWire` 设备驱动。

要确定你希望加载的驱动是否支持 `udev`，可以用模块名字作为参数运行 `modinfo`。然后查看 `/sys/bus` 下的设备目录里是否有个 `modalias` 文件。

如果在 `sysfs` 下能找到 `modalias` 文件，那么就能驱动这个设备并可以直接操作它，但是如果该文件里没有包含设备别名，那意味着这个驱动有问题。我们可以先尝试不依靠 `udev` 直接加载驱动，并寄希望于这个问题能在日后得到解决。

如果在 `/sys/bus` 下的相应目录里没有 `modalias` 的话，意味着内核开发人员还没有为这个总线类型增加 `modalias` 支持。使用 Linux-4.20.12 内核，应该是 `ISA` 总线的问题。希望这个可以在后面的内核版本里得到解决。

`udev` 根本不打算加载 `snd-pcm-oss` 这样的「封装」驱动程序和 `loop` 这样的非硬件设备驱动。

### 7.3.3.2. 内核驱动没有自动加载，udev 也没有尝试加载

如果「封装」模块只是强化其它模块的功能（比如，`snd-pcm-oss` 模块通过允许 `OSS` 应用直接访问声卡的方式加强了 `snd-pcm` 模块的功能），需要配置 `modprobe` 在 `udev` 加载硬件驱动模块后再加载相应的封装模块。为此，可以在对应的 `/etc/modprobe.d/<filename>.conf` 文件中增加「`softdep`」行。例如：

```
softdep snd-pcm post: snd-pcm-oss
```

请注意「`softdep`」也支持 `pre:` 的依赖方式，或者混合 `pre:` 和 `post:`。查看 `modprobe.d(5)` 手册了解更多关于「`softdep`」语法和功能的信息。

如果问题模块并非一个封装，其本身也是有用的话，配置 `modules` 开机脚本在引导系统的时候加载模块。这样需要把模块名字添加到 `/etc/sysconfig/modules` 文件里的单独一行。这也可以用于封装模块，但是仅作为备选方案。

### 7.3.3.3. udev 加载了一些无用模块

要么不要编译该模块，要么把它加入到模块黑名单 `/etc/modprobe.d/blacklist.conf` 里，如下的例子中屏蔽了 `forte` 模块：

```
blacklist forte
```

被屏蔽的模块仍然可以用 `modprobe` 强行加载。



### 7.3.3.4. udev 创建了错误的设备节点，或错误的软链接

这个情况通常是因为设备匹配错误。例如，一条写的不好的规则可能同时匹配到 SCSI 磁盘（希望加载的）和对应厂商的 SCSI 通用设备（错误的）。找出这条问题规则，并通过 `udevadm info` 命令的帮助改得更具体一些。

### 7.3.3.5. udev 规则不能可靠的工作

这可能是上个问题的另一种表现形式。若非如此，而且你的规则使用了 `sysfs` 特性，那可能是内核时序问题，希望在后面版本的内核中能得以解决。目前的话，你可以暂时建立一条规则等待使用的 `sysfs` 特性，并附加到 `/etc/udev/rules.d/10-wait_for_sysfs.rules` 文件里（如果没有这个文件就创建一个）。如果你是这样做的，并且起作用了，请务必通过 LFS 开发邮件列表通知我们。

### 7.3.3.6. udev 没有创建设备

后面的内容会假设驱动已经静态编译进内核或已经作为模块加载，而且你也已经确认 `udev` 没有创建相应的设备节点。

如果内核驱动没有将一个设备的信息导出至 `sysfs` 系统，则 `udev` 无法创建相应的设备结点。这种情况经常会在内核树之外的第三方驱动程序中出现。其解决方法是在文件 `/lib/udev/devices` 中，使用正确的主设备号和次设备号创建一个静态设备结点（相应的设备号可以在内核文档中的 `devices.txt` 文件或者由第三方驱动程序的文档中找到）。之后 `udev` 会根据这些信息在 `/dev` 中创建一个静态设备结点。

### 7.3.3.7. 设备名称顺序在重启后随机改变

这是因为 `udev` 被设计成并行处理 `uevents` 并加载模块，所以是不可预期的顺序。这个不会被「修复」。你不应该依赖稳定的内核模块名称。而是在检测到设备的稳定特征，比如序列号或 `udev` 安装的一些 `*_id` 应用的输出，来判断设备的稳定名称，之后创建自己的规则生成相应的软链接。可以参考 Section#7.4, “设备管理”和 Section#7.2, “通用网络配置”。

## 7.3.4. 有用的读物

其他的帮助文档可以参考下面的链接：

- `devfs` 的用户空间实现：[http://www.kroah.com/linux/talks/ols\\_2003\\_udev\\_paper/Reprint-Kroah-Hartman-OLS2003.pdf](http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf)
- `sysfs` 文件系统：<http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf>

## 7.4. 设备管理

### 7.4.1. 处理相似性质的设备

正如 Section#7.3, “设备与模块管理概述”中解释的，具有相同功能的设备出现在 `/dev` 目录下的顺序是随机的。假如你有一个 USB 摄像头和一个电视调谐器，`/dev/video0` 有可能是 USB 摄像头，`/dev/video1` 是电视调谐器，有时候又可能是反过来的。对于除声卡和网卡外的设备，都可以通过创建自定义持久性符号链接的 `udev` 规则来固定。网卡如何设置请看 Section#7.2, “通用网络配置”，网卡的相关设置请看 BLFS。

对于你所有的硬件，都有可能遇到此问题（尽管此问题可能在你当前的 Linux 发行版上不存在），在 `/sys/class` 或是 `/sys/block` 目录下找到对应目录，比如，显卡可能的路径为 `/sys/class/video4linux/videoX`。找到该设备的唯一设备标识（通常，厂商和产品 ID 以及/或 序列号会有用）：

```
udevadm info -a -p /sys/class/video4linux/video0
```

然后通过写入规则建立符号链接：

```
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
```

最终，`/dev/video0` 和 `/dev/video1` 依旧会随机分配给 USB 摄像头和电视调谐器，但是 `/dev/tvtuner` 和 `/dev/webcam` 将会固定的分配给正确的设备。

## 7.5. 配置系统时间

本节将会讨论如何配置 `systemd-timedated` 系统服务，包括配置系统时间和时区。

如果你不确定是否将硬件时钟设置为 UTC，可以通过 `hwclock --localtime --show` 来查看。这将根据硬件时钟显示当前的时间。如果显示的和手表的时间相同，应该是设置为本地时间了；如果 `hwclock` 输出的时间不一致，应该是设置为 UTC 了。通过增减数小时，可以确定你所在时区。

`systemd-timedated` 读取 `/etc/adjtime`，然后确定是本地时间还是 UTC 时间。

如果想要将硬件时钟设置为本地时间，使用以下命令建立 `/etc/adjtime`：

```
cat > /etc/adjtime << "EOF"
0.0 0 0.0
0
LOCAL
EOF
```

如果第一次启动时 `/etc/adjtime` 文件不存在，`systemd-timedated` 会认为硬件时钟设置成 UTC 并且以此调整该文件。

你也可以使用 `timedatectl` 程序来告诉 `systemd-timedated` 你的硬件时钟是 UTC 还是本地时间：

```
timedatectl set-local-rtc 1
```

`timedatectl` 也可以用来更改系统时间和时区。

要更改当前系统时间，使用下面的命令：

```
timedatectl set-time YYYY-MM-DD HH:MM:SS
```

硬件时钟也会相应更新。

要更改当前时区，使用下面命令：

```
timedatectl set-timezone TIMEZONE
```

你可以通过运行下面命令查看可用时区列表：

```
timedatectl list-timezones
```

### Note

`timedatectl` 仅可用于 `systemd` 环境。

### 7.5.1. 网络时间同步

从版本号 213 的 `systemd` 开始，包括了一个名为 `systemd-timesyncd` 的守护进程，将能够和远程的 NTP 服务器同步时间。

此守护进程并不是为了取代已有的 NTP 服务，而是作为 SNTP 协议的实现客户端，它可用于更高级的任务和资源有限的系统。

从版本号 216 的 `systemd` 开始，`systemd-timesyncd` 守护进程默认启用，如果需要禁用，执行以下命令：

```
systemctl disable systemd-timesyncd
```

`systemd-timesyncd` 也可以通过配置 `/etc/systemd/timesyncd.conf` 文件同步更改 NTP 服务。

需要注意的是，当使用本地时间作为系统时间时，`systemd-timesyncd` 无法更新硬件时钟。

## 7.6. 配置 Linux 控制台

(译者注：对于中文用户，因为大部分用户的键盘布局都是标准 us 键盘，所以大可忽略此文件的配置)

本节将会讨论如何配置 `systemd-vconsole-setup` 系统服务，包括配置控制台字体和控制台的键盘映射。

`systemd-vconsole-setup` 服务读取 `/etc/vconsole.conf` 中的配置信息，确定使用的键盘类型和屏幕的字体。<http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html> 页面有很多其它语言的「HOWTO」内容可以给你很多帮助。`localectl list-keymaps` 可以列举出所有可用的终端键盘布局。`/usr/share/consolefonts` 目录提供了所有可用的字体。

`/etc/vconsole.conf` 的每一行都应该形如：`VARIABLE="value"`。以下列举了可用的 `VARIABLE`：

**KEYMAP**

此变量指定了键盘的按键映射表。如未设置，默认为 `us`。

**KEYMAP\_TOGGLE**

此变量指定配置第二个切换键盘映射，默认不设置。

**FONT**

此变量指定虚拟控制台的字体。

**FONT\_MAP**

此变量指定要使用的控制台映射。

**FONT\_UNIMAP**

此变量指定 Unicode 字体映射。

以下为德语键盘和和控制台的实例：

```
cat > /etc/vconsole.conf << "EOF"
KEYMAP=de-latin1
FONT=Lat2-Terminus16
EOF
```

你可以使用 `localectl` 实用程序更改 `KEYMAP` 值：

```
localectl set-keymap MAP
```

## Note

`localectl` 仅可用于 `systemd` 环境。

`localectl` 实用程序跟随相应的参数也可以更改 X11 的键盘布局、模型、变形和选项：

```
localectl set-x11-keymap LAYOUT [MODEL] [VARIANT] [OPTIONS]
```

下面的 `localectl` 命令列出了 `localectl set-x11-keymap` 命令参数的可用值：

`list-x11-keymap-models`

显示所有已知的 X11 键盘映射模型。

`list-x11-keymap-layouts`

显示所有已知的 X11 键盘映射布局。

`list-x11-keymap-variants`

显示所有已知的 X11 键盘映射变形。

`list-x11-keymap-options`

显示所有已知的 X11 键盘映射选项。

## Note

想要使用以上的参数，你需要从 BLFS 中安装 Xkeyboard 软件包。

## 7.7. 配置系统语言环境

本地语言的支持依赖于 `/etc/locale.conf`，它包含不少和此相关的环境变量。更改此文件后，可能会出现以下的变化：

- 程序的输出将以本地语言展示
- 修正字符在字母、数字等类型的分类。对于非英语区域设置来说，只有这样，`bash` 才能正常显示非 ASCII 字符
- 国家顺序可以按照字母顺序正常排序（译者注：这里所谓的正常排序，应该是首先将国家名称转换成本地语言的国家名称，比如中国的开头字母是 Z，那么显示就很靠后了。但是假如显示为英文，就是 C 开头，那么将会靠前显示。）
- 默认纸张尺寸
- 货币、时间和日期值的格式

`/etc/locale.conf` 中 `<ll>` 使用语言代码代替（比如中文是「zh」，英文是「en」），`<cc>` 使用地区代码代替（比如中国是「CN」，中国香港是「HK」，中国台湾是「TW」，美国是「US」），`<charmap>` 使用选定字符集指定的标准字符映射表替换。诸如「@euro」这样的可选修饰符也可使用。

运行以下命令可以获得当前 Glibc 支持的本地字符集。

```
locale -a
```

字符映射表可能存在很多的别名，比如「ISO-8859-1」可以写作「iso8859-1」或「iso88591」。但是有一些程序不支持这些乱七八糟的写法（比如「UTF-8」只能写作「UTF-8」，「utf8」它就不认识了）。所以，为了安全起见，在设置的时候还是尽量地使用特定区域设置的规范名称。可以通过以下命令查询在特定区域下的字符映射表标准名称，`<locale name>` 为运行 `locale -a` 输出的首选区域设置（这里以「zh\_CN.utf8」为例）。

```
LC_ALL=<locale name> locale charmap
```

对于「zh\_CN.utf8」以上命令将会如下输出：

```
UTF-8
```

根据以上输出，我们再次修改 `/etc/locale.conf`，将字符映射表设置为标准形式（「zh\_CN.utf8」变为「zh\_CN.UTF-8」）。同理，也可一并查询以下设置的标准命令，然后将其添加到 `bash` 的启动文件中（译者注：对于 `bash` 来说，启动文件为 `.bashrc`）。

```
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
```

以上的命令将会打印当前区域设置的语言、字符编码、本地货币单位以及电话国际编码。如果出现类似下文的错误输出，可能是你没有严格按照第六章指导的方法操作或者是你当前所用的 Glibc 不支持。

```
locale: Cannot set LC_* to default locale: No such file or directory
```

如果这种情况真的发生，你应该使用 `localedef` 命令安装对应的系统区域，或者是考虑更改为其它的区域。假如没有出现错误提示，我们就可以继续进行下一步操作了！

有一些 LFS 之外的包可能出现对你设置的区域支持很差劲的情况。比如 X 的库（X Windows System 的一部分），就可能在内部文件中输出以下消息：

```
Warning: locale not supported by Xlib, locale set to C
```

在若干情况下，Xlib 希望以带规范破折号的大写形式列出字符映射表。比如「ISO-8859-1」而不应该写作「iso88591」。不过，也可以通过去除区域规范中的字符映射部分找到合适的规范。这可以通过运行 `locale charmap` 命令来检查。例如，需要更改「de\_DE.ISO-8859-15@euro」为「de\_DE@euro」以便 Xlib 能识别区域。

即便如此，也可能遇到某些程序因为区域设置和它们预置的不同而导致功能异常（可能不会显示任何的错误消息）。如果出现这样的情况，可以通过查看其它的发行版是如何进行设置区域，从而得到启发。

一旦确定了到底该使用哪个区域设置，就可以创建 `/etc/locale.conf` 文件了：

```
cat > /etc/locale.conf << "EOF"
LANG=<ll>_<CC>.<charmap><@modifiers>
EOF
```

也可以通过 `systemd` 提供的实用程序 `localectl` 修改 `/etc/locale.conf`：

```
localectl set-locale LANG="<ll>_<CC>.<charmap><@modifiers>"
```

此处，也可以指定其它和语言相关的环境变量，例如 `LANG`，`LC_CTYPE`，`LC_NUMERIC` 或任何其它 `locale` 输出的环境变量，只需要用一个空格分开它们就可以了。以下示例中，`LANG` 设置为 `en_US.UTF-8`，但 `LC_CTYPE` 设置为 `en_US`：

```
localectl set-locale LANG="en_US.UTF-8" LC_CTYPE="en_US"
```

## Note

`localectl` 仅可用于 `systemd` 环境。

「C」（默认）和「en\_US」（推荐美国英语用户使用）这两种区域设置有所不同。「C」使用 US-ASCII 7 位字符集，并把设置了最高位的字节作为无效字符。这就是为什么类似 `ls` 的命令本地化时会用疑问号代替。同样，如果你想要使用 `Mutt` 或 `Pine` 发送包含有类似字符的邮件，将会得到如下消息：非 RFC 兼容字符（发送邮件中的字符集为「unknown 8-bit」）。所以，如果你一定以及肯定一定不会用到 8 位的字符，那你可以仅使用「C」。

不少程序还不支持 UTF-8 区域设置。我们正在完善文档并修复类似问题，可以查看<http://www.linuxfromscratch.org/blfs/view/8.4/introduction/locale-issues.html>。

## 7.8. Creating the /etc/inputrc File

The `inputrc` file is the configuration file for the Readline library, which provides editing capabilities while the user is entering a line from the terminal. It works by translating keyboard inputs into specific actions. Readline is used by Bash and most other shells as well as many other applications.

Most people do not need user-specific functionality 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. Creating the /etc/shells File

The `shells` file contains a list of login shells on the system. Applications use this file to determine whether a shell is valid. For each shell a single line should be present, consisting of the shell's path, relative to the root of the directory structure (/).

For example, this file is consulted by `chsh` to determine whether an unprivileged user may change the login shell for her own account. If the command name is not listed, the user will be denied of change.

It is a requirement for applications such as GDM which does not populate the face browser if it can't find `/etc/shells`, or FTP daemons which traditionally disallow access to users with shells not included in this file.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

## 7.10. systemd Usage and Configuration

### 7.10.1. Basic Configuration

The `/etc/systemd/system.conf` file contains a set of options to control basic systemd operations. The default file has all entries commented out with the default settings indicated. This file is where the log level may be changed as well as some basic logging settings. See the `systemd-system.conf(5)` manual page for details on each configuration option.

### 7.10.2. Disabling Screen Clearing at Boot Time

The normal behavior for systemd is to clear the screen at the end of the boot sequence. If desired, this behavior may be changed by running the following command:

```
mkdir -pv /etc/systemd/system/getty@tty1.service.d

cat > /etc/systemd/system/getty@tty1.service.d/noclear.conf << EOF
[Service]
TTYVTDisallocate=no
EOF
```

The boot messages can always be reviewed by using the `journalctl -b` command as the root user.

### 7.10.3. Disabling tmpfs for /tmp

By default, `/tmp` is created as a tmpfs. If this is not desired, it can be overridden by the following:

```
ln -sfv /dev/null /etc/systemd/system/tmp.mount
```

Alternatively, if a separate partition for `/tmp` is desired, specify that partition in an `/etc/fstab` entry.

#### Warning

Do not create the symbolic link above if a separate partition is used for `/tmp`. This will prevent the root file system (`/`) from being remounted `r/w` and make the system unusable when booted.

### 7.10.4. Configuring Automatic File Creation and Deletion

There are several services that create or delete files or directories:

- `systemd-tmpfiles-clean.service`
- `systemd-tmpfiles-setup-dev.service`
- `systemd-tmpfiles-setup.service`

The system location for the configuration files is `/usr/lib/tmpfiles.d/*.conf`. The local configuration files are in `/etc/tmpfiles.d`. Files in `/etc/tmpfiles.d` override files with the same name in `/usr/lib/tmpfiles.d`. See `tmpfiles.d(5)` manual page for file format details.

Note that the syntax for the `/usr/lib/tmpfiles.d/*.conf` files can be confusing. For example, the default deletion of files in the `/tmp` directory is located in `/usr/lib/tmpfiles.d/tmp.conf` with the line:

```
q /tmp 1777 root root 10d
```

The type field, q, discusses creating a subvolume with quotas which is really only applicable to btrfs filesystems. It references type v which in turn references type d (directory). This then creates the specified directory if it is not present and adjusts the permissions and ownership as specified. Contents of the directory will be subject to time based cleanup if the age argument is specified.

If the default parameters are not desired, then the file should be copied to `/etc/tmpfiles.d` and edited as desired. For example:

```
mkdir -p /etc/tmpfiles.d
cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d
```

## 7.10.5. Overriding Default Services Behavior

The parameter of a unit can be overridden by creating a directory and a configuration file in `/etc/systemd/system`. For example:

```
mkdir -pv /etc/systemd/system/foobar.service.d

cat > /etc/systemd/system/foobar.service.d/foobar.conf << EOF
[Service]
Restart=always
RestartSec=30
EOF
```

See `systemd.unit(5)` manual page for more information. After creating the configuration file, run `systemctl daemon-reload` and `systemctl restart foobar` to activate the changes to a service.

## 7.10.6. Debugging the Boot Sequence

Rather than plain shell scripts used in SysVinit or BSD style init systems, systemd uses a unified format for different types of startup files (or units). The command `systemctl` is used to enable, disable, control state, and obtain status of unit files. Here are some examples of frequently used commands:

- `systemctl list-units -t <service> [--all]`: lists loaded unit files of type service.
- `systemctl list-units -t <target> [--all]`: lists loaded unit files of type target.
- `systemctl show -p Wants <multi-user.target>`: shows all units that depend on the multi-user target. Targets are special unit files that are analogous to runlevels under SysVinit.
- `systemctl status <servicename.service>`: shows the status of the servicename service. The `.service` extension can be omitted if there are no other unit files with the same name, such as `.socket` files (which create a listening socket that provides similar functionality to `inetd/xinetd`).

## 7.10.7. Working with the systemd Journal

Logging on a system booted with systemd is handled with `systemd-journald` (by default), rather than a typical unix `syslog` daemon. You can also add a normal `syslog` daemon and have both work side by side if desired. The `systemd-journald` program stores journal entries in a binary format rather than a plain text log file. To assist with parsing the file, the command `journalctl` is provided. Here are some examples of frequently used commands:

- `journalctl -r`: shows all contents of the journal in reverse chronological order.
- `journalctl -u UNIT`: shows the journal entries associated with the specified UNIT file.
- `journalctl -b[=ID] -r`: shows the journal entries since last successful boot (or for boot ID) in reverse chronological order.
- `journalctl -f`: provides functionality similar to `tail -f` (follow).

## 7.10.8. Long Running Processes

Beginning with `systemd-230`, all user processes are killed when a user session is ended, even if `nohup` is used, or the process uses the `daemon()` or `setsid()` functions. This is a deliberate change from a historically permissive environment to a more restrictive one. The new behavior may cause issues if you depend on long running programs (e.g., `screen` or `tmux`) to remain active after ending your user session. There are three ways to enable lingering processes to remain after a user session is ended.

- Enable process lingering for only selected users: Normal users have permission to enable process lingering with the command `loginctl enable-linger` for their own user. System administrators can use the same command with a *user* argument to enable for a user. That user can then use the `systemd-run` command to start long running processes. For example: `systemd-run --scope --user /usr/bin/screen`. If you enable lingering for your user, the `user@.service` will remain even after all login sessions are closed, and will automatically start at system boot. This has the advantage of explicitly allowing and disallowing processes to run after the user session has ended, but breaks backwards compatibility with tools like `nohup` and utilities that use `daemon()`.
- Enable system-wide process lingering: You can set `KillUserProcesses=no` in `/etc/logind.conf` to enable process lingering globally for all users. This has the benefit of leaving the old method available to all users at the expense of explicit control.
- Disable at build-time: You can enable lingering by default while building `systemd` by adding the switch `-Ddefault-kill-user-processes=no` to the `meson` command for `systemd`. This completely disables the ability of `systemd` to kill user processes at session end.



## Chapter 8. 让 LFS 系统可引导

### 8.1. 简介

是时候该让 LFS 系统可以启动了。本章节将讨论以下内容：创建 `fstab` 文件、为新的 LFS 系统编译内核、安装 GRUB 引导器。如此，就可以在电脑启动的时候选择启动 LFS 系统了。

### 8.2. 创建 `/etc/fstab` 文件

`/etc/fstab` 文件的作用是让其它程序确定存储设备的默认挂载点、挂载参数和检查信息（例如完整性检测）。仿照以下格式新建一个文件系统列表（file system table，简称 `fstab`）文件：

```
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

# End /etc/fstab
EOF
```

其中，`<xxx>`，`<yyy>` 和 `<fff>` 请使用适当的值替换。例如 `sda2`，`sda5` 和 `ext4`。关于文件中六个字段的含义，请查看 `man 5 fstab`（译者注：fsck 列的数值来决定需要检查的文件系统的检查顺序。允许的数字是 0，1，和 2。根目录应当获得最高的优先权 1，其它所有需要被检查的设备设置为 2。0 表示设备不会被 fsck 所检查）。

基于 MS-DOS 或者是来源于 Windows 的文件系统（例如：`vfat`，`ntfs`，`smbfs`，`cifs`，`iso9660`，`udf`）需要在挂载选项中添加「`iocharset`」，才能让非 ASCII 字符的文件名正确解析。此选项的值应该与语言区域设置的值相同，以便让内核能正确处理。此选项在相关字符集定义已为内核内建或是编译为模块时生效（在文件系统 -> 本地语言支持中查看）。此外，`vfat` 和 `smbfs` 还需启用「`codepage`」支持。例如，想要挂载 USB 闪存设备，`zh-CN.GBK` 用户需要在 `/etc/fstab` 中添加以下的挂载选项：

```
noauto,user,quiet,showexec,iocharset=gbk,codepage=936
```

对于 `zh_CN.UTF-8` 用户的对应选项是：

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=936
```

需要注意的是，`iocharset` 默认值是 `iso8859-1`（其保证文件系统大小写敏感），而 `utf8` 这个参数告知内核使用 UTF-8 转换文件名，以便可以在 UTF-8 语言环境中解释它们。

此外，还有可能在内核的配置过程中就指定一些文件系统的默认代码页和 `iocharset` 值。相关参数有「默认 NLS 选项」（`CONFIG_NLS_DEFAULT`），「默认远程 NLS 选项」（`CONFIG_SMB_NLS_DEFAULT`），「FAT 默认代码页」（`CONFIG_FAT_DEFAULT_CODEPAGE`），和「FAT 默认 IO 字符集」（`CONFIG_FAT_DEFAULT_IOCHARSET`）。不过，无法在内核编译阶段指定 `ntfs` 文件系统的设置。

另外，一些硬盘类型在遇到电源故障时，假如在 `/etc/fstab` 中使用 `barrier=1` 这个挂载选项，则会让 `ext3` 文件系统的数据更加安全。如需检查磁盘是否支持此选项，请运行 `hdparm`。例如：

```
hdparm -I /dev/sda | grep NCQ
```

如果有输出内容，则代表选项可用。

注意：基于 逻辑卷管理 (LVM) 的分区不可使用 `barrier` 选项。

## 8.3. Linux-4.20.12

Linux 软件包包含 Linux 内核。

大致构建用时: 4.4 - 66.0 SBU (通常约为 6 SBU)  
 所需磁盘空间: 960 - 4250 MB (通常约为 1100 MB)

### 8.3.1. 安装内核

编译内核包括以下步骤——配置、编译和安装。阅读内核源码树中的 README 可以获得替代本手册配置的方法。

运行以下命令准备编译：

```
make mrproper
```

这将保证内核树的绝对干净。内核小组建议在每次编译之前都执行此命令，无用的代码将会在解压后删除。

通过菜单界面配置内核。配置内核的一般信息请查看：<http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt>。BLFS 包含有一些内核的特殊配置，可以查看：<http://www.linuxfromscratch.org/blfs/view/8.4/longindex.html#kernel-config-index>。内核配置和编译的附加信息可查看：<http://www.kroah.com/lkn/>。

#### Note

配置内核的一个好的起点是运行 `make defconfig`。这样会参考你的机器架构生成一份基本能用的基础配置。

注意要确保启用/禁用/设置下面这些特性，否则系统也许不能正常工作甚至根本无法启动：

```
General setup -->
  [ ] Enable deprecated sysfs features to support old userspace tools [CONFIG_SYSFS_DEPRECATED]
  [ ] Enable deprecated sysfs features by default [CONFIG_SYSFS_DEPRECATED_V2]
  [*] open by fhandle syscalls [CONFIG_FHANDLE]
  [ ] Auditing support [CONFIG_AUDIT]
  [*] Control Group support [CONFIG_CGROUPS]
Processor type and features --->
  [*] Enable seccomp to safely compute untrusted bytecode [CONFIG_SECCOMP]
Networking support --->
  Networking options --->
    <*> The IPv6 protocol [CONFIG_IPV6]
Device Drivers --->
  Generic Driver Options --->
    [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
    [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
    [ ] Fallback user-helper invocation for firmware loading [CONFIG_FW_LOADER_USER_HELPER]
Firmware Drivers --->
  [*] Export DMI identification via sysfs to userspace [CONFIG_DMIID]
File systems --->
  [*] Inotify support for userspace [CONFIG_INOTIFY_USER]
  <*> Kernel automounter version 4 support (also supports v3) [CONFIG_AUTOFS4_FS]
Pseudo filesystems --->
  [*] Tmpfs POSIX Access Control Lists [CONFIG_TMPFS_POSIX_ACL]
  [*] Tmpfs extended attributes [CONFIG_TMPFS_XATTR]
Kernel hacking --->
  Choose kernel unwinder (Frame pointer unwinder) ---> [CONFIG_UNWINDER_FRAME_POINTER]
```

#### Note

尽管 "The IPv6 Protocol" 不是必须打开，但是 systemd 开发人员强烈推荐打开。

## Note

如果你主机的硬件用的是 UEFI，那么上面的‘make defconfig’应该会自动添加一些 EFI 相关的内核选项。

为了让你的 LFS 内核，在你的主机是 UEFI 引导环境的情况下，能够被引导，你的内核必须要有这项：

```
Processor type and features --->
[*]   EFI stub support [CONFIG_EFI_STUB]
```

文件 lfs-uefi.txt 中包含了管理 UEFI 环境的完整描述，参见 <http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt>。

上述配置项的一些原理说明：

*Support for uevent helper*

打开这个选项会影响 udev/Eudev 设备管理。

*Maintain a devtmpfs*

这个选项允许内核在 udev 运行之前就创建自动设备节点。之后 udev 在这个基础上运行，管理权限以及增加软链接。对于所有 udev/Eudev 用户，这个配置项是必须的。

```
make menuconfig
```

可选 make 环境变量的含义：

`LANG=<host_LANG_value> LC_ALL=`

建立与宿主系统相同的地域设定。在 UTF-8 linux 文本命令行上逐行绘制适宜的 menuconfig ncurses 接口时可能需要这项配置。

要使用的话，请我务必使用宿主系统中的变量 `$LANG` 去代替 `<host_LANG_value>`。你也可以用宿主系统中的 `$LC_ALL` 或 `$LC_CTYPE` 来代替。

另外，make oldconfig 在某些情况下可能更合适。查看 README 文件了解更多信息。

想偷懒的话，可以拷贝宿主系统的内核配置文件 `.config` (如果有的话) 到解压后的 `linux-4.20.12` 目录下跳过头配置。不过，我们不建议这样做。最好是探索一下整个内核配置菜单，从最开始配置内核。

编译内核映像和模块：

```
make
```

如果使用内核模块，需要 `/etc/modprobe.d` 文件里的模块配置。关于模块和内核配置的信息可以查看 Section#7.3, “设备与模块管理概述”以及 `linux-4.20.12/Documentation` 目录下的内核文档。还有，`modprobe.d(5)` 也可以看一下。

如果内核配置里用到，需要安装模块：

```
make modules_install
```

在内核编译完成后，还需要一个额外步骤来完成安装。有些文件需要拷贝到 `/boot` 目录下。

## Caution

如果宿主系统拥有单独的 `/boot` 分区，那么文件就应该复制到那里。简单的解决方法就是在执行前将 `/boot` 绑定到宿主的 `/mnt/lfs/boot`。以宿主系统中的 root 用户运行：

```
mount --bind /boot /mnt/lfs/boot
```

内核映像文件所在的实际目录根据主机系统架构可能会不一样。下面的文件名你也可以改成你喜欢的，不过开头最好是 `vmlinuz` 才可以兼容下一节要讲的配置引导过程的自动设定。下面的命令假设主机是 x86 架构：

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-4.20.12-lfs-8.4-systemd
```

`System.map` 是内核的符号文件。它映射了每一个内核 API 函数的入口，以及内核运行时的数据结构地址。是调查内核问题时的资源。运行下面的命令安装映射文件：

```
cp -iv System.map /boot/System.map-4.20.12
```

在之前命令 `make menuconfig` 里生成的内核配置文件 `.config` 包含了当前编译的内核的所有配置。最好能保存下来留作参考：

```
cp -iv .config /boot/config-4.20.12
```

安装 Linux 内核文档：

```
install -d /usr/share/doc/linux-4.20.12
cp -r Documentation/* /usr/share/doc/linux-4.20.12
```

需要注意一下内核源代码目录下的文件属主并不是 `root`。在以 `root` 用户解压包的时候（我们在 `chroot` 环境里做的），解压出来的文件会拥有生成这个包的电脑里用户和组。在安装其他包的时候这并不是问题，因为它们的源代码在安装完后就删除了。不过，Linux 内核的源代码经常会保留比较长时间。这样的话，就有可能把软件包作者的用户 ID 对应到本机的某个用户上。从而这个用户就会拥有内核源代码的写权限。

### Note

在很多情况下，内核的配置信息需要在稍后安装来自于 BLFS 的软件包后更新。这和其他的软件包不同，在安装完成编译好的内核后不需要将内核源码树删除。

如果想要保留内核的源码树，在 `linux-4.20.12` 下运行 `chown -R 0:0` 来确保所有文件的所有者都 `root`。

### Warning

一些内核文档里建议创建软链接 `/usr/src/linux` 指向内核源代码目录。这是 2.6 及以前版本内核的特定要求，而在 LFS 系统里一定不要创建这个链接，因为这样的话，在你的基础 LFS 系统完成后安装某些软件包时可能引起问题。

### Warning

系统 `include` 目录 (`/usr/include`) 下的头文件应该总是和编译 Glibc 时用到的头文件保持一致。就是在 Section#6.7, “Linux-4.20.12 API 头文件”里整理过的头文件。因此，它们不要替换成原始内核头文件或任何清理过的内核头文件。

## 8.3.2. 配置 Linux 模块加载顺序

虽然大多数情况下，Linux 模块会被自动加载，但是有时候需要特别指定加载顺序。`modprobe` 或 `insmod` 在加载模块时会读取 `/etc/modprobe.d/usb.conf`。如果将 USB 设备 (`ehci_hcd`、`ohci_hcd` 和 `uhci_hcd`) 编译为模块，则需要此文件，这样它们就会以正确的顺序加载。`ehci_hcd` 需要在 `ohci_hcd` 和 `uhci_hcd` 之前加载，否则在系统启动过程中将会输出警告。

运行以下命令建立 `/etc/modprobe.d/usb.conf` 文件：

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Begin /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# End /etc/modprobe.d/usb.conf
EOF
```

## 8.3.3. Linux 的内容

安装的文件： `config-4.20.12`, `vmlinuz-4.20.12-lfs-8.4-systemd`, and `System.map-4.20.12`  
 安装的目录： `/lib/modules`, `/usr/share/doc/linux-4.20.12`

### 简要说明

`config-4.20.12`

包含内核的所有配置选项

<code>vmlinuz-4.20.12-lfs-8.4-systemd</code>	Linux 系统的引擎。当电脑启动时，内核作为整个系统的第一部分载入。它首先检测和初始化所有的电脑硬件，然后将这些硬件模块抽象成文件树让软件访问，并把单个 CPU 转换成多任务系统，可以看上去同时地运行多个程序
<code>System.map-4.20.12</code>	地址和符号列表；包含有入口点的映射以及所有函数和内核数据结构的地址

## 8.4. Using GRUB to Set Up the Boot Process

### 8.4.1. Introduction

#### Warning

Configuring GRUB incorrectly can render your system inoperable without an alternate boot device such as a CD-ROM. This section is not required to boot your LFS system. You may just want to modify your current boot loader, e.g. Grub-Legacy, GRUB2, or LILO.

Ensure that an emergency boot disk is ready to “rescue” the computer if the computer becomes unusable (un-bootable). If you do not already have a boot device, you can create one. In order for the procedure below to work, you need to jump ahead to BLFS and install **xorriso** from the **libisoburn** package.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

#### Note

To boot LFS on host systems that have UEFI enabled, the kernel needs to have been built with the `CONFIG_EFI_STUB` capability described in the previous section. However, LFS can be booted using GRUB2 without such an addition. To do this, the UEFI Mode and Secure Boot capabilities in the host system's BIOS need to be turned off. For details, see the `lfs-uefi.txt` hint at <http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt>.

### 8.4.2. GRUB Naming Conventions

GRUB uses its own naming structure for drives and partitions in the form of `(hdm)`, where `n` is the hard drive number and `m` is the partition number. The hard drive number starts from zero, but the partition number starts from one for normal partitions and five for extended partitions. Note that this is different from earlier versions where both numbers started from zero. For example, partition `sda1` is `(hd0,1)` to GRUB and `sdb3` is `(hd1,3)`. 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)`.

### 8.4.3. Setting Up the Configuration

GRUB works by writing data to the first physical track of the hard disk. This area is not part of any file system. The programs there access GRUB modules in the boot partition. The default location is `/boot/grub/`.

The location of the boot partition is a choice of the user that affects the configuration. One recommendation is to have a separate small (suggested size is 100 MB) partition just for boot information. That way each build, whether LFS or some commercial distro, can access the same boot files and access can be made from any booted system. If you choose to do this, you will need to mount the separate partition, move all files in the current `/boot` directory (e.g. the linux kernel you just built in the previous section) to the new partition. You will then need to unmount the partition and remount it as `/boot`. If you do this, be sure to update `/etc/fstab`.

Using the current `lfs` partition will also work, but configuration for multiple systems is more difficult.

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 `sda2`.

Install the GRUB files into `/boot/grub` and set up the boot track:

## 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).

```
grub-install /dev/sda
```

## Note

If the system has been booted using UEFI, grub-install will try to install files for the x86\_64-efi target, but those files have not been installed in chapter 6. If this is the case, add `--target i386-pc` to the command above.

## 8.4.4. Creating the GRUB Configuration File

Generate `/boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 4.20.12-lfs-8.4-systemd" {
    linux    /boot/vmlinuz-4.20.12-lfs-8.4-systemd root=/dev/sda2 ro
}
EOF
```

## Note

From GRUB's perspective, the kernel files are relative to the partition used. If you used a separate `/boot` partition, remove `/boot` from the above `linux` line. You will also need to change the `set root` line to point to the boot partition.

GRUB is an extremely powerful program and it provides a tremendous number of options for booting from a wide variety of devices, operating systems, and partition types. There are also many options for customization such as graphical splash screens, playing sounds, mouse input, etc. The details of these options are beyond the scope of this introduction.

## Caution

There is a command, `grub-mkconfig`, that can write a configuration file automatically. It uses a set of scripts in `/etc/grub.d/` and will destroy any customizations that you make. These scripts are designed primarily for non-source distributions and are not recommended for LFS. If you install a commercial Linux distribution, there is a good chance that this program will be run. Be sure to back up your `grub.cfg` file.

## Chapter 9. 尾声

### 9.1. 最后的最后

干的很棒嘛！至此，全新的 LFS 系统就已经安装完成啦！我们衷心地祝愿，你亲手定制的崭新 Linux 系统能陪着你乘风破浪。

创建一个 systemd 所需的 `/etc/os-release` 文件：

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="8.4-systemd"
ID=lfs
PRETTY_NAME="Linux From Scratch 8.4-systemd"
VERSION_CODENAME="<your name here>"
EOF
```

建议和非 systemd 分支一样创建文件 `/etc/lfs-release`。该文件能帮助你（和我们，如果你需要我们帮助的话）确定你当前使用的 LFS 版本。运行以下命令以创建该文件：

```
echo 8.4-systemd > /etc/lfs-release
```

推荐遵守 Linux Standards Base (LSB)，建立文件以显示当前系统的完整信息。运行以下命令新建此文件：

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="8.4-systemd"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

你可以在「`DISTRIB_CODENAME`」字段填写一些特别的字符来彰显「你的」系统的与众不同！

### 9.2. 为 LFS 用户数添砖加瓦

截至此刻，你已经读完了这本书。你想要为 LFS 用户数添砖加瓦吗？赶快点击鼠标访问 <http://www.linuxfromscratch.org/cgi-bin/lfscounter.php> 输入用户名和第一次使用的 LFS 版本注册成为 LFS 用户吧。

赶快重启到 LFS 吧！

### 9.3. 重启系统

至此，所有的软件都已安装完毕，是时候重启你的电脑了。然而，你也应该注意一些事情。通过学习本书建立起来的系统属于最小系统，这也就意味着可能会缺失一些你需要的功能。就是说你还需要做些事情。当重启进入你的新 LFS 中时，这是一个在当前的 chroot 环境中安装一些 BLFS 书中的额外软件包的好时机。以下给出了一些建议：

- 文本模式的浏览器，例如 Lynx，可以在虚拟终端中访问这本 BLFS 书，以进行后续的编译打包工作。
- GPM（GPM：一个支持控制台和 xterm 的鼠标服务）软件包可以让你在虚拟终端中更方便的执行复制/粘贴工作。
- 如果静态 IP 配置不能很好的适用于你当前环境的网络配置，可以安装 `dhcpcd` 或者是 `dhcpc` 的客户端部分来解决。
- 安装 `sudo`，以便在非 root 用户环境下编译软件包，且可以很轻松的在新系统中安装编译出来的软件。
- 如果你想要在舒适的 GUI 环境远程访问新系统，请安装 `openssh` 及其依赖包 `openssl`。
- 为了更加便利的从网络中下载文件，请安装 `wget`。
- 如果你有 GUID 分区表（GPT）类型的磁盘，你也许需要 `gptfdisk` 或是 `parted`。
- 最后，检查以下的配置文件是不是都是正确的吧。
  - `/etc/bashrc`
  - `/etc/dircolors`
  - `/etc/fstab`
  - `/etc/hosts`
  - `/etc/inputrc`



- /etc/profile
- /etc/resolv.conf
- /etc/vimrc
- /root/.bash\_profile
- /root/.bashrc

辛苦了那么久，是该初次启动我们崭新的 LFS 系统的时候了！首先，请退出 chroot 环境：

```
logout
```

然后卸载虚拟文件系统：

```
umount -v $LFS/dev/pts
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

卸载 LFS 文件系统本身：

```
umount -v $LFS
```

如果还建立了其它的挂载点，请在卸载 LFS 文件系统之前先卸载它们：

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

至此，重启系统吧：

```
shutdown -r now
```

这里假设 GRUB 引导器已经如前文所述安装完毕且配置正确，启动项也已经自动设置为 LFS 8.4。

重启后，LFS 便已经可以使用了，你可以安装一些其它的软件以满足自己的需求。

## 9.4. 接下来做什么呢？

十分感谢你耐心的阅读这本 LFS 书，我们十分期待本书能够为你构建系统带来一点点的帮助。

我猜，你现在一定很开心——LFS 系统已经安装完成。「但是，下面该作些什么呢？」不用担心，我们早已经帮你准备好以下资源！

### • 维护

定期检查软件的 bug 和安全公告。因为在从源码构建出 LFS 之后，你便应该养成经常去查看这些报告的好习惯。有关查询的去处，网上倒是有一些不错的资源，这里列举几个：

#### • CERT (计算机应急响应小组)

CERT 有一个邮件列表，专门公示各种操作系统和应用程序的安全警报。订阅信息请点击此链接查看：<http://www.us-cert.gov/cas/signup.html>。

#### • Bugtraq

Bugtraq 是一个专门公示计算机安全的邮件列表。它公示新发现的安全问题，偶尔还会尽可能的提出修补方案。订阅信息请点击此链接查看：<http://www.securityfocus.com/archive>。

### • Beyond Linux From Scratch

Beyond Linux From Scratch (BLFS) 涵盖了比 LFS 书多得多的应用程序。BLFS 项目主页是：<http://www.linuxfromscratch.org/blfs/>。

### • LFS Hints

LFS Hints 是由 LFS 社区的志愿者提交的教育文集。有关信息访问以下网址取得：<http://www.linuxfromscratch.org/hints/list.html>。

### • 邮件列表

有几个 LFS 相关的邮件列表，在你需要的时候可以订阅，也可通过它获得最新的发展动态，对项目作出力所能及的贡献等等。查看 第 1 章 - 邮件列表 可以获得更多的信息。

### • The Linux Documentation Project (TLDP, Linux 文档项目)

Linux 文档项目 (TLDP) 的目标是通过协作来完善 Linux 文档中的所有不足。TLDP 已经完成了大量的 HOWTO、指南和 man 帮助页。它的网站是：<http://www.tldp.org/>。

## Part IV. 附录

# Appendix A. 缩写和术语

ABI	Application Binary Interface (应用程序二进制接口)
ALFS	Automated Linux From Scratch (自动化 LFS)
API	Application Programming Interface (应用程序设计接口)
ASCII	American Standard Code for Information Interchange (美国信息交换标准代码)
BIOS	Basic Input/Output System (基本输入/输出系统)
BLFS	Beyond Linux From Scratch
BSD	Berkeley Software Distribution (伯克利软件发行版)
chroot	change root (更改根目录)
CMOS	Complementary Metal Oxide Semiconductor (互补金属氧化物半导体)
COS	Class Of Service (服务等级)
CPU	Central Processing Unit (中央处理单元)
CRC	Cyclic Redundancy Check (循环冗余码校验)
CVS	Concurrent Versions System (并发版本系统)
DHCP	Dynamic Host Configuration Protocol (动态主机配置协议)
DNS	Domain Name Service (域名服务)
EGA	Enhanced Graphics Adapter (增强型图形适配器)
ELF	Executable and Linkable Format (可执行和可链接格式)
EOF	End of File (文件或数据流结束标志)
EQN	equation (相等)
ext2	second extended file system (第二代可扩展文件系统)
ext3	third extended file system (第三代可扩展文件系统)
ext4	fourth extended file system (第四代可扩展文件系统)
FAQ	Frequently Asked Questions (常见问题)
FHS	Filesystem Hierarchy Standard (文件系统层次结构标准)
FIFO	First-In, First Out (先进先出)
FQDN	Fully Qualified Domain Name (完全合格的域名)
FTP	File Transfer Protocol (文件传输协议)
GB	Gigabytes
GCC	GNU Compiler Collection (GNU 编译器集合)
GID	Group Identifier (组标志符)
GMT	Greenwich Mean Time (格林威治标准时间)
HTML	Hypertext Markup Language (超文本标记语言)
IDE	Integrated Drive Electronics (智能磁盘设备, 集成电路设备)
IEEE	Institute of Electrical and Electronic Engineers (电气与电子工程师学会)
IO	Input/Output (输入/输出)
IP	Internet Protocol (互联网协议)
IPC	Inter-Process Communication (进程间通信)
IRC	Internet Relay Chat (互联网中继聊天)
ISO	International Organization for Standardization (国际标准化组织)
ISP	Internet Service Provider (因特网服务提供者)
KB	Kilobytes
LED	Light Emitting Diode (发光二极管)
LFS	Linux From Scratch
LSB	Linux Standard Base

MB	Megabytes
MBR	Master Boot Record (主引导记录)
MD5	Message Digest 5 (信息摘要算法第五版)
NIC	Network Interface Card (网络接口卡)
NLS	Native Language Support (本地语言支持)
NNTP	Network News Transport Protocol (网络新闻传输协议)
NPTL	Native POSIX Threading Library (本地 POSIX 线程库)
OSS	Open Sound System (开放声音系统)
PCH	Pre-Compiled Headers (预编译头文件)
PCRE	Perl Compatible Regular Expression (Perl 兼容正则表达式)
PID	Process Identifier (进程标志符)
PTY	pseudo terminal (伪终端)
QOS	Quality Of Service (服务质量)
RAM	Random Access Memory (随机存取存储器)
RPC	Remote Procedure Call (远程程序调用)
RTC	Real Time Clock (实时时钟)
SBU	Standard Build Unit (标准编译单位)
SCO	The Santa Cruz Operation (圣克鲁斯操作)
SHA1	Secure-Hash Algorithm 1 (安全哈希算法1)
TLDP	The Linux Documentation Project (Linux 文档项目)
TFTP	Trivial File Transfer Protocol (简单文件传输协议)
TLS	Thread-Local Storage (线性本地存储)
UID	User Identifier (用户标志符)
umask	user file-creation mask (用户文件创建掩码)
USB	Universal Serial Bus (通用串行接口)
UTC	Coordinated Universal Time (通用协调时间)
UUID	Universally Unique Identifier (通用唯一标识符)
VC	Virtual Console (虚拟控制台)
VGA	Video Graphics Array (视频图形阵列)
VT	Virtual Terminal (虚拟终端)

# 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
- Bruce Dubbs <bdubbs@linuxfromscratch.org> – LFS Managing Editor
- Jim Gifford <jim@linuxfromscratch.org> – CLFS Project Co-Leader
- Pierre Labastie <pierre@linuxfromscratch.org> – BLFS Editor and ALFS Lead
- DJ Lucas <dj@linuxfromscratch.org> – LFS and BLFS Editor
- Ken Moffat <ken@linuxfromscratch.org> – BLFS Editor
- 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 until 2008
- Jean-Philippe Mengual <jmengual@linuxfromscratch.org> – French LFS translation project 2008-2016
- Julien Lepiller <jlepillier@linuxfromscratch.org> – French LFS translation project 2017-present
- Anderson Lizardo <lizardo@linuxfromscratch.org> – Portuguese LFS translation project
- Thomas Reitelbach <tr@erdfunkstelle.de> – German LFS translation project
- Anton Maisak <info@linuxfromscratch.org.ru> – Russian LFS translation project
- Elena Shevcova <helen@linuxfromscratch.org.ru> – Russian 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 Baque <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
- Anton Maisak <info@linuxfromscratch.org.ru> – linuxfromscratch.org.ru mirror

## Asian Mirrors

- Satit Phermasawang <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
- Matthew Burgess <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor
- Nathan Coulson <nathan@linuxfromscratch.org> – LFS-Bootscripts Maintainer
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- Jeroen Coumans <jeroen@linuxfromscratch.org> – Website Developer, FAQ Maintainer
- Manuel Canales Esparcia <manuel@linuxfromscratch.org> – LFS/BLFS/HLFS XML and XSL Maintainer
- Alex Groenewoud – LFS Technical Writer
- Marc Heerdink
- Jeremy Huntwork <jhuntwork@linuxfromscratch.org> – LFS Technical Writer, LFS LiveCD Maintainer
- Bryan Kadzban <bryan@linuxfromscratch.org> – LFS Technical Writer
- Mark Hymers
- Seth W. Klein – FAQ maintainer
- Nicholas Leippe <nicholas@linuxfromscratch.org> – Wiki Maintainer
- Anderson Lizardo <lizardo@linuxfromscratch.org> – Website Backend-Scripts Maintainer
- Randy McMurchy <randy@linuxfromscratch.org> – BLFS Project Leader, LFS Editor
- Dan Nicholson <dnicholson@linuxfromscratch.org> – LFS and BLFS Editor
- Alexander E. Patrakov <alexander@linuxfromscratch.org> – LFS Technical Writer, LFS Internationalization Editor, LFS Live CD Maintainer
- Simon Perreault
- Scot Mc Pherson <scot@linuxfromscratch.org> – LFS NNTP Gateway Maintainer
- Douglas R. Reno <renodr@linuxfromscratch.org> – Systemd Editor
- Ryan Oliver <ryan@linuxfromscratch.org> – CLFS Project Co-Leader
- Greg Schafer <gschafer@zip.com.au> – LFS Technical Writer and Architect of the Next Generation 64-bit-enabling Build Method
- 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 Utlej <jeremy@linuxfromscratch.org> – LFS Technical Writer, Bugzilla Maintainer, LFS-Bootscripts Maintainer
- Zack Winkles <zwinkles@gmail.com> – LFS Technical Writer

# 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, and sometimes four, types of dependencies. The first lists what other packages need to be available in order to compile and install the package in question. The second lists what packages, in addition to those on the first list, need to be available in order to run the test suites. The third 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 hard code 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.

The last list of dependencies are optional packages that are not addressed in LFS, but could be useful to the user. These packages may have additional mandatory or optional dependencies of their own. For these dependencies, the recommended practice is to install them after completion of the LFS book and then go back and rebuild the LFS package. In several cases, re-installation is addressed in BLFS.

## Acl

安装必选依赖:	Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
测试套件依赖:	Automake, Diffutils, Findutils, and Libtool
必须预先安装:	Coreutils, Sed, Tar, and Vim
可供选择依赖:	None

## Attr

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
测试套件依赖:	Automake, Diffutils, Findutils, and Libtool
必须预先安装:	Acl and Libcap
可供选择依赖:	None

## Autoconf

安装必选依赖:	Bash, Coreutils, Grep, M4, Make, Perl, Sed, and Texinfo
测试套件依赖:	Automake, Diffutils, Findutils, GCC, and Libtool
必须预先安装:	Automake
可供选择依赖:	Emacs

## Automake

安装必选依赖:	Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, and Texinfo
测试套件依赖:	Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, and Tar
必须预先安装:	None
可供选择依赖:	None

## Bash

安装必选依赖:	Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, and Texinfo
测试套件依赖:	Shadow
必须预先安装:	None
可供选择依赖:	Xorg

## Bc

安装必选依赖:	Bash, Binutils, Bison, Coreutils, GCC, Glibc, Grep, Make, and Readline
测试套件依赖:	Gawk
必须预先安装:	Linux Kernel
可供选择依赖:	None



## Binutils

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo and Zlib
测试套件依赖:	DejaGNU and Expect
必须预先安装:	None
可供选择依赖:	None

## Bison

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, and Sed
测试套件依赖:	Diffutils, Findutils, and Flex
必须预先安装:	Kbd and Tar
可供选择依赖:	Doxygen (test suite)

## Bzip2

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, and Patch
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Check

安装必选依赖:	GCC, Grep, Make, Sed, and Texinfo
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Coreutils

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed, and Texinfo
测试套件依赖:	Diffutils, E2fsprogs, Findutils, Shadow, and Util-linux
必须预先安装:	Bash, Diffutils, Eudev, Findutils, and Man-DB
可供选择依赖:	Perl Expect and IO:Tty modules (for test suite)

## DejaGNU

安装必选依赖:	Bash, Coreutils, Diffutils, GCC, Grep, Make, and Sed
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Diffutils

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Perl
必须预先安装:	None
可供选择依赖:	None

## E2fsprogs

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, and Util-linux
测试套件依赖:	Procps-ng and Psmisc
必须预先安装:	None
可供选择依赖:	None

## Eudev

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, and Sed
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Expat

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, and Sed
测试套件依赖:	None
必须预先安装:	XML::Parser
可供选择依赖:	None

## Expect

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, and Tcl
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## File

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Zlib
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Findutils

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	DejaGNU, Diffutils, and Expect
必须预先安装:	None
可供选择依赖:	None

## Flex

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, and Texinfo
测试套件依赖:	Bison and Gawk
必须预先安装:	IPRoute2, Kbd, and Man-DB
可供选择依赖:	None

## Gawk

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed, and Texinfo
测试套件依赖:	Diffutils
必须预先安装:	None
可供选择依赖:	None

## Gcc

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, and Texinfo
测试套件依赖:	DejaGNU, Expect, and Shadow
必须预先安装:	None
可供选择依赖:	GNAT and ISL

## GDBM

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, and Sed
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Gettext

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Diffutils, Perl, and Tcl
必须预先安装:	Automake
可供选择依赖:	None

## Glibc

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Python, Sed, and Texinfo
测试套件依赖:	File
必须预先安装:	None
可供选择依赖:	None

## GMP

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed, and Texinfo
测试套件依赖:	None
必须预先安装:	MPFR and GCC
可供选择依赖:	None

## Gperf

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, and Make
测试套件依赖:	Diffutils and Expect
必须预先安装:	None
可供选择依赖:	None

## Grep

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, and Texinfo
测试套件依赖:	Gawk
必须预先安装:	Man-DB
可供选择依赖:	Pcre

## Groff

安装必选依赖:	Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, and Texinfo
测试套件依赖:	No test suite available
必须预先安装:	Man-DB and Perl
可供选择依赖:	GPL Ghostscript

## GRUB

安装必选依赖:	Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, and Xz
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Gzip

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Diffutils and Less
必须预先安装:	Man-DB
可供选择依赖:	None

## Iana-Etc

安装必选依赖:	Coreutils, Gawk, and Make
测试套件依赖:	No test suite available
必须预先安装:	Perl
可供选择依赖:	None

## Inetutils

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, and Zlib
测试套件依赖:	No test suite available
必须预先安装:	Tar
可供选择依赖:	None

## Intltool

安装必选依赖:	Bash, Gawk, Glibc, Make, Perl, Sed, and XML::Parser
测试套件依赖:	Perl
必须预先安装:	None
可供选择依赖:	None

## IProute2

安装必选依赖:	Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, and Linux API Headers
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## Kbd

安装必选依赖:	Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, and Sed
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## Kmod

安装必选依赖:	Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Sed, Xz-Uutils, and Zlib
测试套件依赖:	No test suite available
必须预先安装:	Eudev
可供选择依赖:	None

## Less

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
测试套件依赖:	No test suite available
必须预先安装:	Gzip
可供选择依赖:	Pcre

## Libcap

安装必选依赖:	Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make, and Sed
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	Linux-PAM

## Libelf

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, and Make
测试套件依赖:	No test suite available
必须预先安装:	Linux Kernel
可供选择依赖:	None

## Libffi

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Make, and Sed
测试套件依赖:	DejaGnu
必须预先安装:	Python
可供选择依赖:	None

## Libpipeline

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Check
必须预先安装:	Man-DB
可供选择依赖:	None

## Libtool

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Autoconf, Automake, and Findutils
必须预先安装:	None
可供选择依赖:	None

## Linux Kernel

安装必选依赖:	Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl, and Sed
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## M4

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Diffutils
必须预先安装:	Autoconf and Bison
可供选择依赖:	libsigsegv

## Make

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Perl and Procs-ng
必须预先安装:	None
可供选择依赖:	None

## Man-DB

安装必选依赖:	Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, and Xz
测试套件依赖:	Util-linux
必须预先安装:	None
可供选择依赖:	None

## Man-Pages

安装必选依赖:	Bash, Coreutils, and Make
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## Meson

安装必选依赖:	Ninja and Python
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## MPC

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed, and Texinfo
测试套件依赖:	None
必须预先安装:	GCC
可供选择依赖:	None

## MPFR

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed, and Texinfo
测试套件依赖:	None
必须预先安装:	Gawk and GCC
可供选择依赖:	None

## Ncurses

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, and Sed
测试套件依赖:	No test suite available
必须预先安装:	Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, and Vim
可供选择依赖:	None

## Ninja

安装必选依赖:	Binutils, Coreutils, Gcc, and Python
测试套件依赖:	None
必须预先安装:	Meson
可供选择依赖:	Asciidoc, Doxygen, Emacs, and re2c

## Openssl

安装必选依赖:	Binutils, Coreutils, Gcc, Make, and Perl
测试套件依赖:	None
必须预先安装:	Linux
可供选择依赖:	None

## Patch

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, and Sed
测试套件依赖:	Diffutils
必须预先安装:	None
可供选择依赖:	Ed

## Perl

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, and Zlib
测试套件依赖:	lana-Etc and Procps-ng
必须预先安装:	Autoconf
可供选择依赖:	None

## Pkg-config

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt, and Sed
测试套件依赖:	None
必须预先安装:	Kmod
可供选择依赖:	None

## Popt

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, and Make
测试套件依赖:	Diffutils and Sed
必须预先安装:	Pkg-config
可供选择依赖:	None

## Procps-ng

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Make, and Ncurses
测试套件依赖:	DejaGNU
必须预先安装:	None
可供选择依赖:	None

## Psmisc

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, and Sed
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## Python

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, and Sed
测试套件依赖:	GDB and Valgrind
必须预先安装:	Ninja
可供选择依赖:	Berkeley DB, OpenSSL, SQLite, and Tk

## Readline

安装必选依赖:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, and Texinfo
测试套件依赖:	No test suite available
必须预先安装:	Bash and Gawk
可供选择依赖:	None

## Sed

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, and Texinfo
测试套件依赖:	Diffutils and Gawk
必须预先安装:	E2fsprogs, File, Libtool, and Shadow
可供选择依赖:	None

## Shadow

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, and Sed
测试套件依赖:	No test suite available
必须预先安装:	Coreutils
可供选择依赖:	Acl, Attr, Cracklib, and PAM

## Syslogd

安装必选依赖:	Binutils, Coreutils, GCC, Glibc, Make, and Patch
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## systemd

安装必选依赖:	Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Glibc, Gperf, Grep, Intltool, Libcap, Make, Sed, and Util-linux
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Sysvinit

安装必选依赖:	Binutils, Coreutils, GCC, Glibc, Make, and Sed
测试套件依赖:	No test suite available
必须预先安装:	None
可供选择依赖:	None

## Tar

安装必选依赖:	Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, and Texinfo
测试套件依赖:	Autoconf, Diffutils, Findutils, Gawk, and Gzip
必须预先安装:	None
可供选择依赖:	None

## Tcl

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, and Sed
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None

## Texinfo

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, and Sed
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	None



## Util-linux

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, and Zlib
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	Libcap-ng

## Vim

安装必选依赖:	Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, and Sed
测试套件依赖:	None
必须预先安装:	None
可供选择依赖:	Xorg, GTK+2, LessTif, Python, Tcl, Ruby, and GPM

## XML::Parser

安装必选依赖:	Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make, and Perl
测试套件依赖:	Perl
必须预先安装:	Intltool
可供选择依赖:	None

## Xz

安装必选依赖:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, and Make
测试套件依赖:	None
必须预先安装:	Eudev, GRUB, Kmod, and Man-DB
可供选择依赖:	None

## Zlib

安装必选依赖:	Bash, Binutils, Coreutils, GCC, Glibc, Make, and Sed
测试套件依赖:	None
必须预先安装:	File, Kmod, Perl, and Util-linux
可供选择依赖:	None

# Appendix D. 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.

## D.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/>.

## D.2. The MIT License

Copyright © 1999-2019 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

- Acl: 111
- Attr: 110
- Autoconf: 130
- Automake: 131
- Bash: 119
  - tools: 47
- Bash: 119
  - tools: 47
- Bc: 91
- Binutils: 92
  - tools, pass 1: 31
  - tools, pass 2: 39
- Binutils: 92
  - tools, pass 1: 31
  - tools, pass 2: 39
- Binutils: 92
  - tools, pass 1: 31
  - tools, pass 2: 39
- Bison: 116
  - tools: 48
- Bison: 116
  - tools: 48
- Bzip2: 105
  - tools: 49
- Bzip2: 105
  - tools: 49
- Check: 148
- Coreutils: 144
  - tools: 50
- Coreutils: 144
  - tools: 50
- D-Bus: 177
- DejaGNU: 44
- Diffutils: 149
  - tools: 51
- Diffutils: 149
  - tools: 51
- E2fsprogs: 185
- Expat: 123
- Expect: 43
- File: 88
  - tools: 52
- File: 88
  - tools: 52
- Findutils: 151
  - tools: 53
- Findutils: 151
  - tools: 53
- Flex: 117
- Gawk: 150
  - tools: 54
- Gawk: 150
  - tools: 54
- GCC: 101
  - tools, libstdc++: 38
- tools, pass 1: 33
- tools, pass 2: 40
- GCC: 101
  - tools, libstdc++: 38
- tools, pass 1: 33
- tools, pass 2: 40
- GCC: 101
  - tools, libstdc++: 38
- tools, pass 1: 33
- tools, pass 2: 40
- GCC: 101
  - tools, libstdc++: 38
- tools, pass 1: 33
- tools, pass 2: 40
- GDBM: 121
- Gettext: 135
  - tools: 55
- Gettext: 135
  - tools: 55
- Glibc: 79
  - tools: 36
- Glibc: 79
  - tools: 36
- GMP: 94
- Gperf: 122
- Grep: 118
  - tools: 56
- Grep: 118
  - tools: 56
- Groff: 152
- GRUB: 154
- Gzip: 157
  - tools: 57
- Gzip: 157
  - tools: 57
- iana-Etc: 115
- Inetutils: 124
- Intltool: 129
- IPRoute2: 158
- Kbd: 160
- Kmod: 134
- Less: 156
- Libcap: 112
- Libelf: 137
- libffi: 138
- Libpipeline: 162
- Libtool: 120
- Linux: 204
  - API headers: 77
  - tools, API headers: 35
- Linux: 204
  - API headers: 77
  - tools, API headers: 35
- Linux: 204
  - API headers: 77
  - tools, API headers: 35
- M4: 90
  - tools: 45
- M4: 90
  - tools: 45
- Make: 163

tools: 58  
 Make: 163  
   tools: 58  
 Man-DB: 165  
 Man-pages: 78  
 Meson: 143  
 MPC: 97  
 MPFR: 96  
 Ncurses: 108  
   tools: 46  
 Ncurses: 108  
   tools: 46  
 Ninja: 142  
 OpenSSL: 139  
 Patch: 164  
   tools: 59  
 Patch: 164  
   tools: 59  
 Perl: 126  
   tools: 60  
 Perl: 126  
   tools: 60  
 Pkgconfig: 107  
 Procps-ng: 179  
 Psmisc: 114  
 Python  
   tools: 61  
 python: 140  
 Readline: 89  
 Sed: 113  
   tools: 62  
 Sed: 113  
   tools: 62  
 Shadow: 98  
   configuring: 99  
 Shadow: 98  
   configuring: 99  
 systemd: 173  
 Tar: 167  
   tools: 63  
 Tar: 167  
   tools: 63  
 Tcl: 42  
 Texinfo: 168  
   tools: 64  
 Texinfo: 168  
   tools: 64  
 udev  
   用法: 193  
 Util-linux: 181  
   tools: 65  
 Util-linux: 181  
   tools: 65  
 Vim: 170  
 XML::Parser: 128  
 Xz: 132  
   tools: 66  
 Xz: 132  
   tools: 66  
 Zlib: 87

## Programs

2to3: 140  
 accessdb: 165, 166  
 aclocal: 131, 131  
 aclocal-1.16: 131, 131  
 addftinfo: 152, 152  
 addpart: 181, 182  
 addr2line: 92, 93  
 afmtodit: 152, 152  
 agetty: 181, 182  
 apropos: 165, 166  
 ar: 92, 93  
 as: 92, 93  
 attr: 110, 110  
 autoconf: 130, 130  
 autoheader: 130, 130  
 autom4te: 130, 130  
 automake: 131, 131  
 automake-1.16: 131, 131  
 autopoint: 135, 135  
 autoreconf: 130, 130  
 autoscan: 130, 130  
 autoupdate: 130, 130  
 awk: 150, 150  
 badblocks: 185, 186  
 base64: 144, 145, 144, 145  
 base64: 144, 145, 144, 145  
 basename: 144, 145  
 bash: 119, 119  
 bashbug: 119, 119  
 bc: 91, 91  
 bison: 116, 116  
 blkdiscard: 181, 182  
 blkid: 181, 182  
 blockdev: 181, 182  
 bootctl: 173, 175  
 bridge: 158, 158  
 bunzip2: 105, 105  
 busctl: 173, 175  
 bzip2: 105, 105  
 bzcat: 105, 105  
 bzcmp: 105, 105  
 bzdiff: 105, 105  
 bzegrep: 105, 105  
 bzfgrep: 105, 105  
 bzgrep: 105, 105  
 bzip2: 105, 105  
 bzip2recover: 105, 106  
 bzless: 105, 106  
 bzip2: 105, 105  
 bzip2: 105, 105  
 c++: 101, 103  
 c++filt: 92, 93  
 cal: 181, 182  
 capsh: 112, 112  
 captinfo: 108, 109  
 cat: 144, 145  
 catchsegv: 79, 83  
 catman: 165, 166  
 cc: 101, 103  
 cfdisk: 181, 182  
 chacl: 111, 111

chage:	98, 99	dumpe2fs:	185, 186
chattr:	185, 186	dumpkeys:	160, 161
chcon:	144, 145	e2freefrag:	185, 186
chcpu:	181, 182	e2fsck:	185, 186
checkmk:	148, 148	e2image:	185, 186
chem:	152, 152	e2label:	185, 186
chfn:	98, 99	e2undo:	185, 186
chgpaswd:	98, 99	e4defrag:	185, 186
chgrp:	144, 145	echo:	144, 146
chmod:	144, 145	egrep:	118, 118
choom:	181, 182	eject:	181, 182
chown:	144, 145	elfedit:	92, 93
chpasswd:	98, 99	enc2xs:	126, 127
chroot:	144, 145	encguess:	126, 127
chrt:	181, 182	env:	144, 146
chsh:	98, 99	envsubst:	135, 135
chvt:	160, 160	eqn:	152, 152
cksum:	144, 145	eqn2graph:	152, 152
clear:	108, 109	ex:	170, 171
cmp:	149, 149	expand:	144, 146
col:	181, 182	expect:	43, 43
colcrt:	181, 182	expiry:	98, 99
colrm:	181, 182	expr:	144, 146
column:	181, 182	factor:	144, 146
comm:	144, 145	faillog:	98, 100
compile_et:	185, 186	fallocate:	181, 182
coredumpctl:	173, 175	false:	144, 146
corelist:	126, 127	fdformat:	181, 182
cp:	144, 145	fdisk:	181, 182
cpan:	126, 127	fgconsole:	160, 161
cpp:	101, 103	fgrep:	118, 118
csplit:	144, 145	file:	88, 88
ctrlaltdel:	181, 182	filefrag:	185, 186
ctstat:	158, 158	find:	151, 151
cut:	144, 145	findfs:	181, 182
c_rehash:	139, 139	findmnt:	181, 182
date:	144, 145	flex:	117, 117
dbus-cleanup-sockets:	177, 177	flex++:	117, 117
dbus-daemon:	177, 177	flock:	181, 182
dbus-launch:	177, 177	fmt:	144, 146
dbus-monitor:	177, 177	fold:	144, 146
dbus-run-session:	177, 177	free:	179, 179
dbus-send:	177, 178	fsck:	181, 182
dbus-test-tool:	177, 178	fsck.cramfs:	181, 182
dbus-update-activation-environment:	177, 178	fsck.ext2:	185, 186
dbus-uuidgen:	177, 178	fsck.ext3:	185, 186
dc:	91, 91	fsck.ext4:	185, 186
dd:	144, 145	fsck.ext4dev:	185, 186
deallocvt:	160, 160	fsck.minix:	181, 182
debugfs:	185, 186	fsfreeze:	181, 182
delpart:	181, 182	fstrim:	181, 182
depmod:	134, 134	ftp:	124, 125
df:	144, 145	fuser:	114, 114
diff:	149, 149	g++:	101, 103
diff3:	149, 149	gawk:	150, 150
dir:	144, 145	gawk-4.2.1:	150, 150
dircolors:	144, 145	gcc:	101, 103
dirname:	144, 146	gc-ar:	101, 104
dmesg:	181, 182	gc-nm:	101, 104
dnsdomainname:	124, 124	gc-ranlib:	101, 104
du:	144, 146	gcov:	101, 104



gdbmtool: 121, 121  
gdbm\_dump: 121, 121  
gdbm\_load: 121, 121  
gdiffmk: 152, 152  
gencat: 79, 83  
genl: 158, 158  
getcap: 112, 112  
getconf: 79, 83  
getent: 79, 83  
getfacl: 111, 111  
getfattr: 110, 110  
getkeycodes: 160, 161  
getopt: 181, 182  
getpcaps: 112, 112  
gettext: 135, 135  
gettext.sh: 135, 135  
gettextize: 135, 135  
glilypond: 152, 152  
gpasswd: 98, 100  
gperf: 122, 122  
gperl: 152, 152  
gpinyin: 152, 152  
gprof: 92, 93  
grap2graph: 152, 152  
grep: 118, 118  
gm: 152, 152  
grodvi: 152, 152  
groff: 152, 152  
groffer: 152, 152  
grog: 152, 152  
grolbp: 152, 152  
grolj4: 152, 152  
gropdf: 152, 152  
grops: 152, 152  
grotty: 152, 152  
groupadd: 98, 100  
groupdel: 98, 100  
groupmems: 98, 100  
groupmod: 98, 100  
groups: 144, 146  
grpck: 98, 100  
grpconv: 98, 100  
grpunconv: 98, 100  
grub-bios-setup: 154, 154  
grub-editenv: 154, 154  
grub-file: 154, 154  
grub-fstest: 154, 154  
grub-glue-efi: 154, 154  
grub-install: 154, 154  
grub-kbdcomp: 154, 154  
grub-macbless: 154, 154  
grub-menulst2cfg: 154, 154  
grub-mkconfig: 154, 154  
grub-mkimage: 154, 154  
grub-mklayout: 154, 155  
grub-mknetdir: 154, 155  
grub-mkpasswd-pbkdf2: 154, 155  
grub-mkrelpath: 154, 155  
grub-mkrescue: 154, 155  
grub-mkstandalone: 154, 155  
grub-ofpathname: 154, 155  
grub-probe: 154, 155  
grub-reboot: 154, 155  
grub-render-label: 154, 155  
grub-script-check: 154, 155  
grub-set-default: 154, 155  
grub-setup: 154, 155  
grub-syslinux2cfg: 154, 155  
gunzip: 157, 157  
gzexe: 157, 157  
gzip: 157, 157  
h2ph: 126, 127  
h2xs: 126, 127  
halt: 173, 175  
head: 144, 146  
hexdump: 181, 183  
hostid: 144, 146  
hostname: 124, 125  
hostnamectl: 173, 175  
hpftodit: 152, 153  
hwclock: 181, 183  
i386: 181, 183  
iconv: 79, 83  
iconvconfig: 79, 83  
id: 144, 146  
idle3: 140  
ifcfg: 158, 158  
ifconfig: 124, 125  
ifnames: 130, 130  
ifstat: 158, 158  
indxbib: 152, 153  
info: 168, 168  
infocmp: 108, 109  
infotocap: 108, 109  
init: 173, 175  
insmod: 134, 134  
install: 144, 146  
install-info: 168, 168  
instmodsh: 126, 127  
intltool-extract: 129, 129  
intltool-merge: 129, 129  
intltool-prepare: 129, 129  
intltool-update: 129, 129  
intltoolize: 129, 129  
ionice: 181, 183  
ip: 158, 158  
ipcmk: 181, 183  
ipcrm: 181, 183  
ipcs: 181, 183  
isosize: 181, 183  
join: 144, 146  
journalctl: 173, 175  
json\_pp: 126, 127  
kbdinfo: 160, 161  
kbdrate: 160, 161  
kbd\_mode: 160, 161  
kernel-install: 173, 175  
kill: 181, 183  
killall: 114, 114  
kmod: 134, 134  
last: 181, 183  
lastb: 181, 183

lastlog:	98, 100	makedb:	79, 84
ld:	92, 93	makeinfo:	168, 168
ld.bfd:	92, 93	man:	165, 166
ld.gold:	92, 93	mandb:	165, 166
ldattach:	181, 183	manpath:	165, 166
ldconfig:	79, 83	mapscrn:	160, 161
ldd:	79, 83	mcookie:	181, 183
lddlibc4:	79, 83	md5sum:	144, 146
less:	156, 156	mesg:	181, 183
lessecho:	156, 156	meson:	143, 143
lesskey:	156, 156	mesonconf:	143, 143
lex:	117, 117	mesonintrospect:	143, 143
lexgrog:	165, 166	mesontest:	143, 143
lfskernel-4.20.12:	204, 207	mkdir:	144, 146
libasan:	101, 104	mke2fs:	185, 186
libnetcfg:	126, 127	mkfifo:	144, 146
libtool:	120, 120	mkfs:	181, 183
libtoolize:	120, 120	mkfs.bfs:	181, 183
link:	144, 146	mkfs.cramfs:	181, 183
linux32:	181, 183	mkfs.ext2:	185, 186
linux64:	181, 183	mkfs.ext3:	185, 186
lkbib:	152, 153	mkfs.ext4:	185, 186
ln:	144, 146	mkfs.ext4dev:	185, 186
lnstat:	158, 158	mkfs.minix:	181, 183
loadkeys:	160, 161	mklost+found:	185, 186
loadunimap:	160, 161	mknod:	144, 146
locale:	79, 83	mkswap:	181, 183
localectl:	173, 175	mktemp:	144, 146
localedef:	79, 83	mk_cmds:	185, 186
locate:	151, 151	mmroff:	152, 153
logger:	181, 183	modinfo:	134, 134
login:	98, 100	modprobe:	134, 134
loginctl:	173, 175	more:	181, 183
logname:	144, 146	mount:	181, 183
logoutd:	98, 100	mountpoint:	181, 183
logsave:	185, 186	msgattrib:	135, 135
look:	181, 183	msgcat:	135, 135
lookbib:	152, 153	msgcmp:	135, 135
losetup:	181, 183	msgcomm:	135, 135
ls:	144, 146	msgconv:	135, 135
lsattr:	185, 186	msgen:	135, 135
lsblk:	181, 183	msgexec:	135, 135
lscpu:	181, 183	msgfilter:	135, 135
lsipc:	181, 183	msgfmt:	135, 136
lslocks:	181, 183	msggrep:	135, 136
lslogins:	181, 183	msginit:	135, 136
lsmod:	134, 134	msgmerge:	135, 136
lzcat:	132, 132	msgunfmt:	135, 136
lzcmp:	132, 132	msguniq:	135, 136
lzdiff:	132, 132	mtrace:	79, 84
lzegrep:	132, 132	mv:	144, 146
lzfgrep:	132, 132	namei:	181, 183
lzgrep:	132, 132	ncursesw6-config:	108, 109
lzless:	132, 132	neqn:	152, 153
lzma:	132, 132	networkctl:	173, 175
lzmadec:	132, 132	newgidmap:	98, 100
lzmainfo:	132, 132	newgrp:	98, 100
lzmore:	132, 132	newuidmap:	98, 100
m4:	90, 90	newusers:	98, 100
machinectl:	173, 175	ngettext:	135, 136
make:	163, 163	nice:	144, 146

ninja:	142, 142	prlimit:	181, 183
nl:	144, 146	prove:	126, 127
nm:	92, 93	prtstat:	114, 114
nohup:	144, 146	ps:	179, 179
nologin:	98, 100	psfaddtable:	160, 161
nproc:	144, 146	psfgettable:	160, 161
nroff:	152, 153	psfstriptable:	160, 161
nscd:	79, 84	psfxtable:	160, 161
nsenter:	181, 183	pstree:	114, 114
nstat:	158, 158	pstree.x11:	114, 114
numfmt:	144, 146	ptar:	126, 127
objcopy:	92, 93	ptardiff:	126, 127
objdump:	92, 93	ptargrep:	126, 127
od:	144, 146	ptx:	144, 146
openssl:	139, 139	pwck:	98, 100
openvt:	160, 161	pwconv:	98, 100
partx:	181, 183	pwd:	144, 146
passwd:	98, 100	pwdx:	179, 179
paste:	144, 146	pwunconv:	98, 100
patch:	164, 164	pydoc3:	140
pathchk:	144, 146	python3:	140
pdfmom:	152, 153	pyenv:	140
pdfroff:	152, 153	ranlib:	92, 93
pdftexi2dvi:	168, 168	raw:	181, 183
peekfd:	114, 114	readelf:	92, 93
perl:	126, 127	readlink:	144, 146
perl5.28.1:	126, 127	readprofile:	181, 183
perlbug:	126, 127	realpath:	144, 146
perldoc:	126, 127	reboot:	173, 175
perlivp:	126, 127	recode-sr-latin:	135, 136
perlthanks:	126, 127	refer:	152, 153
pfbtops:	152, 153	rename:	181, 183
pg:	181, 183	renice:	181, 183
pgrep:	179, 179	reset:	108, 109
pic:	152, 153	resize2fs:	185, 186
pic2graph:	152, 153	resizepart:	181, 183
piconv:	126, 127	rev:	181, 183
pidof:	179, 179	rm:	144, 147
ping:	124, 125	rmdir:	144, 147
ping6:	124, 125	rmmod:	134, 134
pinky:	144, 146	roff2dvi:	152, 153
pivot_root:	181, 183	roff2html:	152, 153
pkg-config:	107, 107	roff2pdf:	152, 153
pkill:	179, 179	roff2ps:	152, 153
pl2pm:	126, 127	roff2text:	152, 153
pldd:	79, 84	roff2x:	152, 153
pmap:	179, 179	routef:	158, 158
pod2html:	126, 127	routel:	158, 158
pod2man:	126, 127	rtacct:	158, 158
pod2texi:	168, 169	rtcwake:	181, 183
pod2text:	126, 127	rtmon:	158, 158
pod2usage:	126, 127	rtpr:	158, 159
podchecker:	126, 127	rtstat:	158, 159
podselect:	126, 127	runcon:	144, 147
post-grohtml:	152, 153	runlevel:	173, 175
poweroff:	173, 175	runtest:	44, 44
pr:	144, 146	rview:	170, 171
pre-grohtml:	152, 153	rvim:	170, 171
preconv:	152, 153	script:	181, 184
printenv:	144, 146	scriptreplay:	181, 184
printf:	144, 146	sdiff:	149, 149

sed: 113, 113  
 seq: 144, 147  
 sefacl: 111, 111  
 setarch: 181, 184  
 setcap: 112, 112  
 setfattr: 110, 110  
 setfont: 160, 161  
 setkeycodes: 160, 161  
 settled: 160, 161  
 setmetamode: 160, 161  
 setsid: 181, 184  
 setterm: 181, 184  
 setvtrgb: 160, 161  
 sfdisk: 181, 184  
 sg: 98, 100  
 sh: 119, 119  
 sha1sum: 144, 147  
 sha224sum: 144, 147  
 sha256sum: 144, 147  
 sha384sum: 144, 147  
 sha512sum: 144, 147  
 shasum: 126, 127  
 showconsolefont: 160, 161  
 showkey: 160, 161  
 shred: 144, 147  
 shuf: 144, 147  
 shutdown: 173, 175  
 size: 92, 93  
 slabtop: 179, 179  
 sleep: 144, 147  
 sln: 79, 84  
 soelim: 152, 153  
 sort: 144, 147  
 sotruss: 79, 84  
 splain: 126, 127  
 split: 144, 147  
 sprof: 79, 84  
 ss: 158, 159  
 stat: 144, 147  
 stdbuf: 144, 147  
 strings: 92, 93  
 strip: 92, 93  
 stty: 144, 147  
 su: 98, 100  
 sulogin: 181, 184  
 sum: 144, 147  
 swapon: 181, 184  
 swapoff: 181, 184  
 swapon: 181, 184  
 switch\_root: 181, 184  
 sync: 144, 147  
 sysctl: 179, 179  
 systemctl: 173, 175  
 systemd-analyze: 173, 175  
 systemd-ask-password: 173, 175  
 systemd-cat: 173, 175  
 systemd-cgls: 173, 175  
 systemd-cgtop: 173, 175  
 systemd-delta: 173, 175  
 systemd-detect-virt: 173, 175  
 systemd-escape: 173, 175  
 systemd-hwdb: 173, 175  
 systemd-inhibit: 173, 175  
 systemd-machine-id-setup: 173, 175  
 systemd-mount: 173, 176  
 systemd-notify: 173, 176  
 systemd-nspawn: 173, 176  
 systemd-path: 173, 176  
 systemd-resolve: 173, 176  
 systemd-run: 173, 176  
 systemd-socket-activate: 173, 176  
 systemd-tmpfiles: 173, 176  
 systemd-tty-ask-password-agent: 173, 176  
 tabs: 108, 109  
 tac: 144, 147  
 tail: 144, 147  
 tailf: 181, 184  
 talk: 124, 125  
 tar: 167, 167  
 taskset: 181, 184  
 tbl: 152, 153  
 tc: 158, 159  
 tcsh: 42, 42  
 tcsh8.6: 42, 42  
 tee: 144, 147  
 telinit: 173, 176  
 telnet: 124, 125  
 test: 144, 147  
 texi2dvi: 168, 169  
 texi2pdf: 168, 169  
 texi2any: 168, 169  
 texindex: 168, 169  
 tfmtodit: 152, 153  
 tftp: 124, 125  
 tic: 108, 109  
 timedatectl: 173, 176  
 timeout: 144, 147  
 tload: 179, 179  
 toe: 108, 109  
 top: 179, 179  
 touch: 144, 147  
 tput: 108, 109  
 tr: 144, 147  
 traceroute: 124, 125  
 troff: 152, 153  
 true: 144, 147  
 truncate: 144, 147  
 tset: 108, 109  
 tsort: 144, 147  
 tty: 144, 147  
 tune2fs: 185, 186  
 tzselect: 79, 84  
 udevadm: 173, 176  
 ul: 181, 184  
 umount: 181, 184  
 uname: 144, 147  
 uname26: 181, 184  
 uncompress: 157, 157  
 unexpand: 144, 147  
 unicode\_start: 160, 161  
 unicode\_stop: 160, 161  
 uniq: 144, 147

- unlink: 144, 147
- unlzma: 132, 132
- unshare: 181, 184
- unxz: 132, 132
- updatedb: 151, 151
- uptime: 179, 180
- useradd: 98, 100
- userdel: 98, 100
- usermod: 98, 100
- users: 144, 147
- utmpdump: 181, 184
- uuid: 181, 184
- uuidgen: 181, 184
- vdir: 144, 147
- vi: 170, 171
- view: 170, 171
- vigr: 98, 100
- vim: 170, 171
- vimdiff: 170, 171
- vimtutor: 170, 171
- vipw: 98, 100
- vmstat: 179, 180
- w: 179, 180
- wall: 181, 184
- watch: 179, 180
- wc: 144, 147
- wdctl: 181, 184
- whatis: 165, 166
- whereis: 181, 184
- who: 144, 147
- whoami: 144, 147
- wipefs: 181, 184
- wraptool: 143, 143
- x86\_64: 181, 184
- xargs: 151, 151
- xgettext: 135, 136
- xmlwf: 123, 123
- xsubpp: 126, 127
- xtrace: 79, 84
- xxd: 170, 172
- xz: 132, 132
- xzcat: 132, 132
- xzcmp: 132, 132
- xzdec: 132, 132
- xzdiff: 132, 133
- xzegrep: 132, 133
- xzfgrep: 132, 133
- xzgrep: 132, 133
- xzless: 132, 133
- xzmore: 132, 133
- yacc: 116, 116
- yes: 144, 147
- zcat: 157, 157
- zcmp: 157, 157
- zdiff: 157, 157
- zdump: 79, 84
- zegrep: 157, 157
- zfgrep: 157, 157
- zforce: 157, 157
- zgrep: 157, 157
- zic: 79, 84

- zipdetails: 126, 127
- zless: 157, 157
- zmore: 157, 157
- znew: 157, 157
- zramctl: 181, 184

## Libraries

- Expat: 128, 128
- ld-2.29.so: 79, 84
- libacl: 111, 111
- libanl: 79, 84
- libasprintf: 135, 136
- libattr: 110, 110
- libbftd: 92, 93
- libblkid: 181, 184
- libBrokenLocale: 79, 84
- libbz2: 105, 106
- libc: 79, 84
- libcap: 112, 112
- libcheck: 148, 148
- libcidn: 79, 84
- libcom\_err: 185, 186
- libcrypt: 79, 84
- libcrypto.so: 139, 139
- libcursesw: 108, 109
- libdbus-1: 177, 178
- libdl: 79, 84
- libe2p: 185, 187
- libexpat: 123, 123
- libexpect-5.45: 43, 43
- libext2fs: 185, 187
- libfdisk: 181, 184
- libffi: 138
- libfl: 117, 117
- libformw: 108, 109
- libg: 79, 84
- libgcc: 101, 104
- libgcov: 101, 104
- libgdbm: 121, 121
- libgdbm\_compat: 121, 121
- libgettextlib: 135, 136
- libgettextpo: 135, 136
- libgettextsrc: 135, 136
- libgmp: 94, 95
- libgmpxx: 94, 95
- libgomp: 101, 104
- libhistory: 89, 89
- libiberty: 101, 104
- libieee: 79, 84
- libkmod: 134
- libltdl: 120, 120
- liblto\_plugin: 101, 104
- liblzma: 132, 133
- libm: 79, 84
- libmagic: 88, 88
- libman: 165, 166
- libmandb: 165, 166
- libmcheck: 79, 84
- libmemusage: 79, 84
- libmenuw: 108, 109

libmount: 181, 184	/etc/inputrc: 199
libmpc: 97, 97	/etc/ld.so.conf: 83
libmpfr: 96, 96	/etc/lfs-release: 210
libncursesw: 108, 109	/etc/localtime: 81
libnsl: 79, 84	/etc/lsb-release: 210
libnss: 79, 84	/etc/modprobe.d/usb.conf: 206
libopcodes: 92, 93	/etc/nsswitch.conf: 81
libpanelw: 108, 109	/etc/os-release: 210
libpipeline: 162	/etc/passwd: 73
libprocps: 179, 180	/etc/protocols: 115
libpthread: 79, 84	/etc/resolv.conf: 191
libquadmath: 101, 104	/etc/services: 115
libreadline: 89, 89	/etc/vimrc: 171
libresolv: 79, 84	/usr/include/asm-generic/*.h: 77, 77
librpcsvc: 79, 84	/usr/include/asm/*.h: 77, 77
librt: 79, 84	/usr/include/drm/*.h: 77, 77
libSegFault: 79, 84	/usr/include/linux/*.h: 77, 77
libsmartcols: 181, 184	/usr/include/mtd/*.h: 77, 77
libss: 185, 187	/usr/include/rdma/*.h: 77, 77
libssl.so: 139, 139	/usr/include/scsi/*.h: 77, 77
libssp: 101, 104	/usr/include/sound/*.h: 77, 77
libstdbuf: 144, 147	/usr/include/video/*.h: 77, 77
libstdc++: 101, 104	/usr/include/xen/*.h: 77, 77
libsupc++: 101, 104	/var/log/btmp: 73
libsystemd: 173, 176	/var/log/lastlog: 73
libtcl8.6.so: 42, 42	/var/log/wtmp: 73
libtclstub8.6.a: 42, 42	/var/run/utmp: 73
libthread_db: 79, 84	/etc/locale.conf: 197
libtsan: 101, 104	/etc/shells: 199
libudev: 173, 176	man pages: 78, 78
libutil: 79, 84	systemd Customization: 200
libuuid: 181, 184	
liby: 116, 116	
libz: 87, 87	
preloadable_libintl: 135, 136	

## Scripts

时间  
配置: 196

控制台  
配置: 196

hostname  
configuring: 192

localnet  
/etc/hosts: 192

network  
/etc/hosts: 192  
configuring: 190

network  
/etc/hosts: 192  
configuring: 190

## Others

/boot/config-4.20.12: 204, 206

/boot/System.map-4.20.12: 204, 207

/dev/\*: 69

/etc/fstab: 203

/etc/group: 73

/etc/hosts: 192