Linux从零开始

9.0版

发布于2019年9月1日

由Gerard Beekmans 创建

执行主编：布鲁斯· 杜布斯（Bruce Dubbs）

目录

二。为构建做准备

2.准备主机系统

3.包装和补丁

4.最终准备

5.建立一个临时系统

三，建立LFS系统

6.安装基本系统软件

前言

前言

我学习和更好地了解Linux的旅程始于1998年。我刚刚安装了我的第一个Linux发行版，并很快对Linux背后的整个概念和理念产生了兴趣。

总有很多方法可以完成一项任务。关于Linux发行版也可以这样说。这些年来，存在着很多。有些仍然存在，有些变成了其他东西，而另一些则降级到我们的记忆中。他们为满足目标受众的需求而做的事情有所不同。因为存在许多实现同一最终目标的不同方法，所以我开始意识到，我不再受任何一种实现的限制。在发现Linux之前，由于您别无选择，我们只是忍受了其他操作系统中的问题。无论您是否喜欢它，它都是如此。使用Linux，选择的概念开始出现。如果你不喜欢什么

我尝试了多种发行版，但无法决定任何一种发行版。它们本身就是很棒的系统。不再是非是非。这已成为个人喜好问题。有了所有这些选择，很显然，没有一个系统对我来说是完美的。因此，我着手创建自己的Linux系统，该系统完全符合我的个人喜好。

为了真正使它成为我自己的系统，我决定从源代码编译所有内容，而不是使用预编译的二进制包。这种“ 完美的 ” Linux系统将具有各种系统的优点，而不会出现其缺点。起初，这个想法相当艰巨。我仍然致力于建立这样一个系统的想法。

在解决了循环依赖和编译时错误等问题之后，我终于构建了一个定制的Linux系统。它与当时的任何其他Linux系统一样，完全可操作且完全可用。但这是我自己的创造。我自己建立了这样一个系统，这非常令人满足。唯一更好的办法是自己创建每个软件。这是下一件好事。

当我与Linux社区的其他成员分享我的目标和经验时，很明显，这些想法一直受到人们的关注。这种定制的Linux系统不仅可以满足用户的特定要求，而且还为程序员和系统管理员提供了一个理想的学习机会，以增强他们（现有的）Linux技能，这很快变得显而易见。出于这种广泛兴趣，*Linux From Scratch项目*诞生了。

本书《 Linux From Scratch》是该项目的核心。它提供了设计和构建自己的系统所需的背景和说明。虽然本书提供了可以使系统正常工作的模板，但是您可以自由地更改说明以适合自己，这在一定程度上是该项目的重要组成部分。您保持控制；我们只是伸出援助之手，让您开始自己的旅程。

我衷心希望您能度过美好的时光，在自己的Linux From Scratch系统上工作，并享受拥有真正属于您自己的系统的诸多好处。

-
杰拉德·比克曼斯
杰拉德AT查看LinuxFromScratch D0T组织

听众

您想读这本书的原因有很多。许多人提出的问题之一是：“ 当您只需下载并安装现有的Linux系统时，为什么要经历从头开始手动构建Linux系统的所有麻烦？”

该项目存在的重要原因之一是可以帮助您从内而外学习Linux系统的工作方式。构建LFS系统有助于证明是什么使Linux产生了影响，以及事物如何协同工作并相互依赖。这种学习经验可以提供的最好的事情之一就是能够定制Linux系统以满足您自己的独特需求的能力。

LFS的另一个主要优点是，它使您可以对系统进行更多控制，而不必依赖其他人的Linux实现。使用LFS，您可以坐在驾驶员的座位上，并决定系统的各个方面。

LFS允许您创建非常紧凑的Linux系统。在安装常规发行版时，通常会迫使您安装很多可能从未使用过或不了解的程序。这些程序浪费资源。您可能会争辩说，对于当今的硬盘驱动器和CPU，不再需要考虑这些资源。但是，有时候，如果没有其他因素，您仍然会受到尺寸方面的限制。考虑一下可启动CD，USB记忆棒和嵌入式系统。这些都是LFS可以带来好处的领域。

定制Linux系统的另一个优点是安全性。通过从源代码编译整个系统，您可以审核所有内容并应用所需的所有安全补丁。不再需要等待其他人来编译可修复安全漏洞的二进制程序包。除非您检查补丁并自己实施，否则不能保证新的二进制软件包已正确构建并足以解决此问题。

Linux From Scratch的目标是构建一个完整且可用的基础级系统。如果您不想从头开始构建自己的Linux系统，则可以从本书中受益。

建立您自己的LFS系统还有很多其他充分的理由要在此处列出。最后，教育是迄今为止最有力的理由。随着您继续进行LFS体验，您将发现信息和知识真正带来的力量。

LFS目标架构

LFS的主要目标体系结构是AMD / Intel x86（32位）和x86_64（64位）CPU。另一方面，这本书中的说明也可以通过Power PC和ARM CPU进行一些修改后工作。要构建使用这些CPU之一的系统，除了接下来几页中的内容外，主要的先决条件是现有的Linux系统，例如较早的LFS安装，Ubuntu，Red Hat / Fedora，SuSE或其他发行版本。针对您拥有的架构。另请注意，可以在64位AMD / Intel计算机上安装32位发行版并将其用作主机系统。

需要在此处添加有关64位系统的其他一些事实。与32位系统相比，可执行程序的大小稍大一些，而任意程序的执行速度仅稍快一些。例如，在基于Core2Duo CPU的系统上的LFS-6.5测试版本中，测量了以下统计信息：

```
Architecture Build Time     Build Size
32-bit       198.5 minutes  648 MB
```

```
64-bit      190.6 minutes  709 MB
```

如您所见，64位版本仅快32%，比32位版本快4%。进入64位系统的收益相对较小。当然，如果您拥有超过4GB的RAM或要处理超过4GB的数据，则64位系统的优势是巨大的。

> **注意**
>
> 以上讨论仅在比较基于相同硬件的构建时才适用。现代的64位系统比早期的64位系统快得多，LFS作者建议在有选择的情况下在64位系统上构建。

由LFS生成的默认64位版本被认为是"纯" 64位系统。也就是说，它仅支持64位可执行文件。构建"多库"系统需要两次编译许多应用程序，一次针对32位系统，一次针对64位系统。LFS不直接支持此功能，因为它会干扰提供简单的基本Linux系统所需说明的教育目标。您可以参考" **从头开始跨Linux"**项目中的高级主题。

LFS和标准

LFS的结构尽可能遵循Linux标准。主要标准是：

- **POSIX.1-2008**。
- **文件系统层次结构标准（FHS）3.0版**
- **Linux Standard Base（LSB）版本5.0（2015）**

  LSB有四个独立的标准：核心，桌面，运行时语言和映像。除了通用要求外，还存在特定于体系结构的要求。还有两个可供试用的区域：Gtk3和图形。LFS尝试符合上一节中讨论的体系结构。

  > **注意**
  >
  > 许多人不同意LSB的要求。对其进行定义的主要目的是确保专有软件将能够在兼容的系统上安装并正常运行。由于LFS是基于源的，因此用户可以完全控制所需的软件包，许多用户选择不安装LSB指定的某些软件包。

创建一个能够通过LSB认证测试的完整LFS系统是可能的，但是如果没有许多超出LFS范围的其他软件包，则不可能。这些附加软件包在BLFS中有安装说明。

LFS提供的软件包必须满足LSB要求

LSB核心：	Bash，Bc，Binutils，Coreutils，Diffutils，文件，Findutils，Gawk，Grep，Gzip，M4，Man-DB，Ncurses，Procps，Psmisc，Sed，Shadow，Tar，Util-linux，Zlib
LSB桌面：	没有

LSB运行时语言：	佩尔
LSB成像：	没有
LSB Gtk3和LSB图形（试用版）：	没有

BLFS提供的软件包必须满足LSB要求

LSB核心：	At，Batch（At的一部分），Cpio，Ed，Fcrontab，Initd-tools，Lsb_release，NSPR，NSS，PAM，Pax，Sendmail（或Postfix或Exim），时间
LSB桌面：	Alsa，ATK，Cairo，桌面文件实用程序，Freetype，Fontconfig，Gdk-pixbuf，Glib2，GTK + 2，图标命名实用程序，Libjpeg-turbo，Libpng，Libtiff，Libxml2，MesaLib，Pango，Xdg-utils， Xorg
LSB运行时语言：	Python，Libxml2，Libxslt
LSB成像：	CUPS，杯式过滤器，Ghostscript，SANE
LSB Gtk3和LSB图形（试用版）：	GTK + 3

LFS或BLFS不提供的软件包都需要满足LSB的要求

LSB核心：	没有
LSB桌面：	Qt4（但提供了Qt5）
LSB运行时语言：	没有
LSB成像：	没有
LSB Gtk3和LSB图形（试用版）：	没有

书中包装的基本原理

如前所述，LFS的目标是建立一个完整且可用的基础级系统。这包括复制自身所需的所有软件包，同时提供相对最小的基础，可根据用户的选择从中基础定制更完整的系统。这并不意味着LFS是可能的最小系统。其中包括一些并非严格要求的重要软件包。下面的列表记录了本书中每个软件包的原理。

- Acl

  该软件包包含用于管理访问控制列表的实用程序，这些实用程序用于为文件和目录定义更细粒度的自由访问权限。

- Attr

  该软件包包含用于管理文件系统对象上的扩展属性的程序。

- 自动配置

该软件包包含用于生成shell脚本的程序，该程序可以从开发人员的模板自动配置源代码。在更新构建过程之后，通常需要重新构建软件包。

- 自动制作

  该软件包包含用于从模板生成Make文件的程序。在更新构建过程之后，通常需要重新构建软件包。

- 重击

  该软件包满足了LSB核心要求，即为系统提供Bourne Shell接口。之所以选择它，是因为它的通用用法和超出基本Shell功能的广泛功能，而使其成为其他Shell软件包的首选。

- 公元前

  该软件包提供了任意精度的数值处理语言。它满足构建Linux内核时所需的要求。

- Binutils

  该软件包包含一个链接器，一个汇编器和其他用于处理目标文件的工具。需要使用此软件包中的程序来编译LFS系统及更高版本中的大多数软件包。

- 野牛

  该软件包包含构建其他LFS程序所需的GNU版本的yacc（又是另一个编译器）。

- Bzip2

  该软件包包含用于压缩和解压缩文件的程序。需要解压缩许多LFS软件包。

- 校验

  该软件包包含用于其他程序的测试工具。它仅安装在临时工具链中。

- Coreutils

  该软件包包含许多用于查看和操作文件和目录的基本程序。这些程序对于命令行文件管理是必需的，对于LFS中每个软件包的安装过程都是必需的。

- DejaGNU

  该软件包包含用于测试其他程序的框架。它仅安装在临时工具链中。

- Diffutils

  该程序包包含显示文件或目录之间差异的程序。这些程序可用于创建补丁，也可用于许多软件包的构建过程。

- E2fsprogs

  该软件包包含用于处理ext2，ext3和ext4文件系统的实用程序。这些是Linux支持的最常见且经过全面测试的文件系统。

- 尤德夫

该软件包是一个设备管理器。在系统中添加或删除设备时，它动态控制/ dev目录中的条目。

- 外籍人士

  该软件包包含一个相对较小的XML解析库。XML :: Parser Perl模块需要它。

- 期望

  该软件包包含一个程序，用于与其他交互式程序进行脚本化对话。它通常用于测试其他软件包。它仅安装在临时工具链中。

- 文件

  该软件包包含一个用于确定给定文件或多个文件类型的实用程序。一些软件包需要它来构建。

- Findutils

  该软件包包含用于在文件系统中查找文件的程序。它在许多软件包的构建脚本中使用。

- 柔性

  该软件包包含一个实用程序，用于生成识别文本模式的程序。它是lex（词法分析器）程序的GNU版本。需要构建多个LFS软件包。

- 高克

  该软件包包含用于处理文本文件的程序。它是awk（Aho-Weinberg-Kernighan）的GNU版本。它在许多其他软件包的构建脚本中使用。

- 海湾合作委员会

  该软件包是Gnu编译器集合。它包含C和C ++编译器以及其他不是LFS生成的其他编译器。

- GDBM

  该软件包包含GNU数据库管理器库。其他LFS软件包Man-DB使用它。

- 文字

  该软件包包含实用程序和库，用于许多软件包的国际化和本地化。

- 格里布

  该软件包包含主要的C库。没有它，Linux程序将无法运行。

- GMP

  该软件包包含数学库，这些库为任意精度算术提供有用的功能。需要构建Gcc。

- Gperf

  该程序包包含一个程序，该程序可以从键集中生成完美的哈希函数。Eudev需要它。

- 格列普

  该软件包包含用于搜索文件的程序。大多数程序包的构建脚本都使用这些程序。

- 格罗夫

  该软件包包含用于处理和格式化文本的程序。这些程序的一项重要功能是格式化手册页。

- 格鲁布

  该软件包是Grand Unified Boot Loader。它是可用的几种引导加载程序之一，但最灵活。

- 压缩文件

  该软件包包含用于压缩和解压缩文件的程序。需要解压缩LFS及更高版本中的许多软件包。

- 雅娜等

  该软件包提供了网络服务和协议的数据。需要启用适当的联网功能。

- Inetutils

  该软件包包含用于基本网络管理的程序。

- 国际工具

  该软件包包含用于从源文件中提取可翻译字符串的工具。

- IP路由2

  该软件包包含用于基本和高级IPv4和IPv6网络的程序。由于其IPv6功能，它是在其他通用网络工具包（net-tools）中选择的。

- 千比特

  该软件包包含键表文件，非美国键盘的键盘实用程序以及许多控制台字体。

- 科莫德

  该软件包包含管理Linux内核模块所需的程序。

- 减

  该软件包包含一个非常漂亮的文本文件查看器，允许您在查看文件时向上或向下滚动。Man-DB也使用它来查看手册页。

- Libcap

  该软件包实现了Linux内核中可用的POSIX 1003.1e功能的用户空间接口。

- Libelf

elfutils项目提供了用于ELF文件和DWARF数据的库和工具。该软件包中的大多数实用程序在其他软件包中都可用，但是使用默认（也是最有效的）配置来构建Linux内核需要该库。

- 利比菲

  该软件包为各种调用约定实现了一个可移植的高级编程接口。有些程序在编译时可能不知道要将哪些参数传递给函数。例如，可以在运行时告知解释器用于调用给定函数的参数的数量和类型。Libffi可在此类程序中使用，以提供从解释程序到已编译代码的桥梁。

- 脂质管道

  Libpipeline软件包包含一个库，用于以灵活方便的方式操纵子流程的管道。Man-DB软件包需要它。

- Libtool

  该软件包包含GNU通用库支持脚本。它包装了在一致的可移植界面中使用共享库的复杂性。其他LFS软件包中的测试套件需要它。

- Linux内核

  该软件包是操作系统。它是GNU / Linux环境中的Linux。

- M4

  该软件包包含一个通用的文本宏处理器，可用作其他程序的构建工具。

- 使

  该程序包包含用于指导程序包构建的程序。LFS中几乎每个软件包都需要它。

- 文库

  该软件包包含用于查找和查看手册页的程序。由于具有卓越的国际化功能，因此选择了它而不是手册包。它提供了man程序。

- 手册页

  该软件包包含基本Linux手册页的实际内容。

- 介子

  该软件包提供了用于自动构建软件的软件工具。Meson的主要目标是最大程度地减少软件开发人员花在配置其构建系统上的时间。

- MPC

  该软件包包含用于复数运算的函数。Gcc要求。

- MPFR

  该软件包包含用于多精度算术的函数。Gcc要求。

- 忍者

该软件包包含一个小型构建系统，该系统注重速度。它旨在让其输入文件由更高级别的构建系统生成，并尽可能快地运行构建。

- Ncurses

  该软件包包含用于终端独立处理字符屏幕的库。它通常用于为菜单系统提供光标控制。LFS中的许多软件包都需要它。

- Openssl

  该软件包提供与加密有关的管理工具和库。这些对于为其他软件包（包括Linux内核）提供加密功能很有用。

- 补丁

  该软件包包含一个程序，该程序用于通过应用通常由diff程序创建的*补丁*文件来修改或创建文件。几个LFS软件包的构建过程需要它。

- 佩尔

  该软件包是运行时语言PERL的解释器。几个LFS软件包的安装和测试套件需要它。

- 包配置

  该软件包提供了一个程序，用于返回有关已安装的库或软件包的元数据。

- Procps-NG

  该软件包包含用于监视过程的程序。这些程序对于系统管理很有用，LFS引导脚本也使用这些程序。

- 伪造

  该软件包包含用于显示有关正在运行的进程的信息的程序。这些程序对于系统管理很有用。

- Python 3

  该软件包提供了一种解释性语言，其语言具有强调代码可读性的设计理念。

- Readline

  该软件包是一组提供命令行编辑和历史记录功能的库。它由Bash使用。

- 塞德

  此程序包允许编辑文本，而无需在文本编辑器中打开它。大多数LFS软件包的配置脚本也需要它。

- 阴影

  该软件包包含用于以安全方式处理密码的程序。

- Sysklogd

  该软件包包含用于记录系统消息的程序，例如发生异常事件时由内核或守护进程提供的消息。

- Sysvinit

  该软件包提供了init程序，该程序是Linux系统上所有其他进程的父级。

- 柏油

  该软件包提供了几乎所有LFS中使用的软件包的归档和提取功能。

- Tcl

  该软件包包含LFS软件包中许多测试套件中使用的工具命令语言。它仅安装在临时工具链中。

- Texinfo

  该软件包包含用于读取，写入和转换信息页面的程序。它在许多LFS软件包的安装过程中使用。

- 实用程序

  该软件包包含其他实用程序。其中包括用于处理文件系统，控制台，分区和消息的实用程序。

- Vim

  该软件包包含一个编辑器。选择它是因为它与经典的vi编辑器兼容并且具有大量强大的功能。对于许多用户而言，编辑器是非常个人的选择，并且如果需要，可以替换任何其他编辑器。

- XML ::解析器

  该软件包是与Expat交互的Perl模块。

- XZ实用工具

  该软件包包含用于压缩和解压缩文件的程序。它提供了通常可用的最高压缩率，可用于以XZ或LZMA格式解压缩程序包。

- Zlib

  该软件包包含某些程序使用的压缩和解压缩例程。

先决条件

构建LFS系统并非易事。为了解决问题并正确执行列出的命令，它需要一定水平的Unix系统管理的现有知识。特别是，作为绝对最低要求，您应该已经可以使用命令行（shell）复制或移动文件和目录，列出目录和文件内容以及更改当前目录。还希望您对使用和安装Linux软件有一定的了解。

由于LFS本书*至少*假定*具备*这种基本技能，因此各种LFS支持论坛不太可能在这些领域为您提供很多帮助。您会发现有关此类基础知识的问题可能无法得到解答，或者您只会被引荐给LFS基本预读清单。

在构建LFS系统之前，我们建议您阅读以下内容：

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

  这是在Linux下构建和安装“通用”Unix软件包的综合指南。尽管它是前一段时间编写的，但它仍然提供了构建和安装软件所需的基本技术的完整摘要。

- 从源代码安装的初学者指南 **http://moi.vonos.net/linux/beginners-installing-from-source/**

  本指南很好地总结了从源代码构建软件所需的基本技能和技术。

版式

为了使事情更容易理解，本书中使用了一些印刷约定。本节包含在Linux From Scratch中发现的印刷格式的一些示例。

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

除非周围的文本中另有说明，否则此文本形式的输入应按所看到的完全键入。在说明部分中也使用它来标识正在引用的命令。

在某些情况下，逻辑行将扩展为两条或更多条物理行，并在行尾加反斜杠。

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

请注意，反斜杠后必须立即返回。其他空格字符（如空格或制表符）将导致错误的结果。

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

这种文本形式（固定宽度的文本）显示屏幕输出，通常是发出命令的结果。此格式还用于显示文件名，例如`/etc/ld.so.conf`。

重点

本书将这种文本形式用于多种目的。其主要目的是强调要点或项目。

http://www.linuxfromscratch.org/

此格式用于LFS社区内和外部页面的超链接。它包括HOWTO，下载位置和网站。

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

创建配置文件时使用此格式。第一条命令告诉系统`$LFS/etc/group`从以下各行中键入的内容创建文件，直到遇到序列结束文件（EOF）。因此，通常将整个部分键入，如所见。

<REPLACED TEXT>

此格式用于封装不按所见方式键入或用于复制和粘贴操作的文本。

[OPTIONAL TEXT]

此格式用于封装可选的文本。

passwd(5)

此格式用于引用特定的手册（手册）页面。括号内的数字表示手册中的特定章节。例如，passwd有两个手册页。根据LFS安装说明，这两个手册页将位于/usr/share/man/man1/passwd.1和/usr/share/man/man5/passwd.5。该书在使用passwd(5)时专门指/usr/share/man/man5/passwd.5。man passwd将打印找到匹配“ passwd ”的第一个手册页，该页将为/usr/share/man/man1/passwd.1。对于此示例，您将需要运行man 5 passwd为了读取所引用的特定页面。应该注意的是，大多数手册页在不同的部分中没有重复的页面名称。因此， 人*<program name>*通常就足够了。

结构体

本书分为以下几部分。

第一部分-简介

第一部分解释了有关如何进行LFS安装的一些重要说明。本节还提供有关该书的元信息。

第二部分-准备构建

第二部分描述了如何为构建过程做准备-创建分区，下载软件包以及编译临时工具。

第三部分-建立LFS系统

第三部分指导读者完成LFS系统的构建，一个一个地编译和安装所有软件包，设置启动脚本，以及安装内核。最终的Linux系统是构建其他软件以根据需要扩展系统的基础。本书结尾处有一个易于使用的参考，列出了所有已安装的程序，库和重要文件。

勘误

用于创建LFS系统的软件正在不断更新和增强。LFS手册发行后，可能会提供安全警告和错误修复。要检查此LFS发行版中的软件包版本或说明是否需要进行任何修改以适应安全漏洞或其他错误修复，请 在继续构建之前访问**http://www.linuxfromscratch.org/lfs/errata/9.0/**。您应该注意显示的所有更改，并在构建LFS系统的过程中将其应用到本书的相关部分。

第一部分简介

第1章简介

1.1。如何建立LFS系统

将使用已经安装的Linux发行版（例如Debian，OpenMandriva，Fedora或openSUSE）构建LFS系统。这个现有的Linux系统（主机）将用作提供必要程序（包括编译器，链接器和外壳程序）以构建新系统的起点。在发行版安装期间选择“ 开发 ”选项，以能够访问这些工具。

作为在计算机上安装单独发行版的替代方法，您可能希望使用商业发行版中的LiveCD。

本书的第2章介绍了如何创建新的Linux本机分区和文件系统。在这里将编译和安装新的LFS系统。第3章 介绍了构建LFS系统需要下载哪些软件包和修补程序以及如何将它们存储在新文件系统中。第4章 讨论了合适的工作环境的设置。请仔细阅读第4章 ，因为它解释了在开始学习第5章 及以后的内容之前您需要了解的几个重要问题。

第5章 说明了许多软件包的安装，这些软件包将构成第6章中用于构建实际系统的基本开发套件（或工具链）。需要其中一些软件包来解决循环依赖关系，例如，要编译编译器，您需要编译器。

第5章 还向您展示了如何构建工具链的第一阶段，包括Binutils和GCC（第一阶段基本上意味着将重新安装这两个核心软件包）。下一步是构建C库Glibc。Glibc将由第一遍构建的工具链程序进行编译。然后，将构建第二遍工具链。这次，工具链将与新建的Glibc动态链接。剩下的第5章 使用此第二遍工具链构建软件包。完成此操作后，LFS安装过程将不再依赖主机分发，运行内核除外。

从主机分发中隔离新系统的这种尝试似乎过分。第5.2节“工具链技术说明”中提供了有关为何执行此操作的完整技术说明。

在第6章中，构建了完整的LFS系统。在chroot环境（改变root）程序用来进入一个虚拟环境，并开始一个新的shell，其根目录被设置为LFS分区。这与重新引导并指示内核将LFS分区挂载为根分区非常相似。该系统实际上并不会重新引导，而是使用chroot，因为创建可引导系统需要额外的工作，而这些工作目前还没有必要。主要优点是“ chrooting ”允许您在构建LFS时继续使用主机系统。在等待软件包编译完成时，您可以继续正常使用计算机。

要完成安装，请在第7章中设置基本的系统配置，并在第8章中设置内核和引导加载程序。 第9章包含有关在本书之外继续LFS经验的信息。实施本书中的步骤之后，计算机将准备好重新启动进入新的LFS系统。

简而言之，这是一个过程。以下各章和程序包描述中讨论了有关每个步骤的详细信息。将澄清看似复杂的项目，并且在您踏上LFS冒险之旅时一切都会落到位。

1.2。自上次发行以来的新功能

以下是自本书上一版以来进行的软件包更新的列表。

升级至：

- BC 2.1.3
- 野牛3.4.1

- Bzip2-1.0.8
- Coreutils-8.31
- E2fsprogs-1.45.3
- Eudev-3.2.8
- 外籍人士2.2.7
- 文件-5.37
- 高克5.0.1
- GCC-9.2.0
- Gettext-0.20.1
- Glibc-2.30
- GRUB-2.04
- IPRoute2-5.2.0
- Kbd-2.2.0
- 少于551
- LFS-引导脚本-20190524
- Libcap 2.27
- Libelf-0.177（来自elfutils）
- Linux-5.2.8
- Man-DB-2.8.6.1
- 手册页5.02
- 介子-0.51.1
- Openssl-1.1.1c
- Perl-5.30.0
- Python-3.7.4
- 暗影-4.7

- SysVinit-2.95
- 焦油1.32
- Texinfo-6.6
- Tzdata-2019b
- Util-Linux-2.34
- Vim-8.1.1846

添加：

已移除：

1.3。变更日志

这是《 Linux From Scratch》一书的9.0版，日期为2019年9月1日。如果这本书已有六个月以上的历史，那么可能已经有更新更好的版本。要找出答案，请通过**http://www.linuxfromscratch.org/mirrors.html**检查其中一个镜像。

以下是自本书上一版以来所做的更改的列表。

变更日志条目：

- 2019-09-01
  - [bdubbs]-LFS-9.0发布。
- 2019-08-14
  - [bdubbs]-更新至vim-8.1.1846。修复**#4500**。
  - [bdubbs]-更新至elfutils-0.177。修复 **#4516**。
  - [bdubbs]-更新至gcc-9.2.0。修复**#4514**。
  - [bdubbs]-更新至bc-2.1.3。修复**#4513**。
  - [bdubbs]-更新至man-db-2.8.6.1。修复 **#4512**。
  - [bdubbs]-更新到linux-5.2.8。修复**#4511**。
- 2019-08-04
  - [bdubbs]-通过将包含文件添加到glibc标头来解决linux-5.2引入的问题。
- 2019-08-03
  - [bdubbs]-更新到linux-5.2.5。修复**#4505**。

- [bdubbs]-更新至kbd-2.2.0。修复#4507。
- [bdubbs]-更新至glibc-2.30。修复#4508。
- [bdubbs]-更新至man-pages-5.02。修复 #4509。

- 2019-07-21
  - [bdubbs]-更新到linux-5.2.2。修复#4504。
  - [bdubbs]-更新至bc-2.1.1。修复#4503。
  - [bdubbs]-更新至kbd-2.1.0。修复#4502。
  - [bdubbs]-更新至e2fsprogs-1.45.3。修复 #4501。

- 2019-07-14
  - [bdubbs]-修复了binutils-2.32 gold链接程序的测试。修复#4498。
  - [bdubbs]-更新至tzdata-2019b。修复#4492。
  - [bdubbs]-更新至python3-3.7.4。修复#4496。
  - [bdubbs]-更新至介子0.51.1。修复#4497。
  - [bdubbs]-更新至iproute2-5.2.0。修复 #4495。
  - [bdubbs]-更新至grub-2.04。修复#4494。
  - [bdubbs]-更新到linux-5.2.1。修复#4493。
  - [bdubbs]-更新至bc-2.1.0。修复#4436。
  - [bdubbs]-更新至bzip2-1.0.8。修复#4499。

- 2019-06-29
  - [bdubbs]-在OpenSSL中正确初始化数据结构，以避免valgrind未初始化的值错误。修复#4491。
  - [bdubbs]-更新至介子0.51.0。修复#4483。
  - [bdubbs]-更新到gawk-5.0.1。修复#4486。
  - [bdubbs]-更新至expat-2.2.7。修复#4488。
  - [bdubbs]-更新到linux-5.1.15。修复#4487。
  - [bdubbs]-更新至sysvinit-2.95。修复#4484。
  - [bdubbs]-更新至bzip2-1.0.7。修复#4490。

- 2019-06-24
  - [renodr]-解决了在版本目录中安装Check文档的问题。感谢Ryan Marsaw的报告。通过删除无法识别/未使用的--docdir并在make install命令中将其替换为" docdir ="来解决此问题。

- 2019-06-18
  - [renodr]-更新到linux-5.1.11。解决了SOCK PANIC问题。修复#4485。

- 2019-06-16
  - [bdubbs]-更新至vim-8.1.1535。修复#4482。
  - [bdubbs]-更新至shadow-4.7。修复#4481。
  - [bdubbs]-更新到linux-5.1.10。修复#4478。
  - [bdubbs]-更新至551以下。修复#4477。
  - [bdubbs]-更新至util-linux-2.34。修复 #4462。
  - [bdubbs]-删除涉及/ tools的eudev指令。修复#4480。
- 2019-06-08
  - [renodr]-请确保仅在systemd中可见第6章中构建Util-Linux之前删除符号链接的说明。
- 2019-06-03
  - [bdubbs]-更新到perl-5.30.0。修复#4474。
  - [bdubbs]-更新到linux-5.1.6。修复#4473。
  - [bdubbs]-更新至openssl-1.1.1c。修复 #4476。
  - [bdubbs]-更新至eudev-3.2.8。修复#4472。
  - [bdubbs]-更新至bison-3.4.1。修复#4471。
  - [bdubbs]-更新至e2fsprogs-1.45.2。修复 #4475。
- 2019-05-24
  - [dj]-LFS引导脚本的外观更改。
- 2019-05-19
  - [bdubbs]-更新至man-pages-5.01。修复 #4467。
  - [bdubbs]-更新到linux-5.1.3。修复#4464。
  - [bdubbs]-更新至iproute2-5.1.0。修复 #4467。
  - [bdubbs]-更新至gettext-0.20.1。修复 #4465。
  - [bdubbs]-更新至文件5.37。修复#4469。
  - [bdubbs]-更新至e2fsprogs-1.45.1。修复 #4468。
- 2019-05-03
  - [bdubbs]-更新至gcc-9.1.0。修复#4463。
  - [bdubbs]-更新到linux-5.0.11。修复#4461。
- 2019-04-23
  - [bdubbs]-使用最新版本的gcc进行更改以允许Perl正确构建。

- 2019-04-20
  - [bdubbs]-更新至perl-5.28.2。修复#4460。
  - [bdubbs]-更新至介子0.50.1。修复#4459。
  - [bdubbs]-更新到linux-5.0.9。修复#4458。
  - [bdubbs]-更新至libcap-2.27。修复#4457。
- 2019-04-15
  - [bdubbs]-更新bzip2网址。修复#4450。
  - [bdubbs]-更新至util-linux-2.33.2。修复 #4454。
  - [bdubbs]-更新到linux-5.0.7。修复#4449。
  - [bdubbs]-更新至gawk-5.0.0。修复#4455。
- 2019-03-27
  - [bdubbs]-恢复为介子0.49.2。
- 2019-03-26
  - [bdubbs]-更新到tzdata2019a。修复#4448。
  - [bdubbs]-更新至Python 3.7.3。修复#4447。
- 2019-03-25
  - [bdubbs]-更新至iproute2-5.0.0。修复 #4446。
  - [bdubbs]-更新到linux-5.0.4。修复#4444。
  - [xry111]-在Glibc构建中使用-ffile-prefix-map而不是-isystem和symlinks来简化指令。
- 2019-03-13
  - [xry111]-更新软件包的内容和简短描述。修复#4443。
- 2019-03-12
  - [bdubbs]-更新至介子0.50.0。修复#4442。
  - [bdubbs]-更新至coreutils-8.31。修复 #4441。
  - [bdubbs]-更新到linux-5.0.1。修复#4440。
  - [bdubbs]-更新至man-pages-5.00。修复 #4439。
  - [bdubbs]-更新至e2fsprogs-1.45.0。修复 #4438。
- 2019-03-05
  - [bdubbs]-更新到linux-5.0。修复#4437。

- 2019-03-01
  - [bdubbs]-更新到texinfo-6.6。修复#4427。
  - [bdubbs]-更新至tar-1.32。修复#4431。
  - [bdubbs]-更新至sysvinit-2.94。修复#4432。
  - [bdubbs]-更新至openssl-1.1.1b。修复 #4435。
  - [bdubbs]-更新至gcc-8.3.0。修复#4430。
  - [bdubbs]-更新到linux-4.20.13。修复#4434。
- 2019-03-01
  - [bdubbs]-LFS-8.4发布。

1.4。资源资源

1.4.1。常问问题

如果在LFS系统的构建过程中遇到任何错误，有任何疑问或认为书中有错别字，请首先查阅位于http：//www.linuxfromscratch的“常见问题”（FAQ）。 org / faq /。

1.4.2。邮件列表

该linuxfromscratch.org服务器托管许多用于开发LFS项目的邮件列表。这些列表包括主要的开发和支持列表等。如果FAQ不能解决您遇到的问题，则下一步是在http://www.linuxfromscratch.org/search.html上搜索邮件列表。

有关不同列表的信息，如何订阅，存档位置以及其他信息，请访问 http://www.linuxfromscratch.org/mail.html。

1.4.3。IRC

LFS社区的一些成员在Internet中继聊天（IRC）上提供帮助。使用此支持之前，请确保在LFS常见问题解答或邮件列表档案中尚未回答您的问题。您可以在找到IRC网络irc.freenode.net。支持渠道名为＃LFS-support。

1.4.4。镜像站点

LFS项目具有许多世界范围的镜像，可以更轻松地访问网站和下载所需的软件包。请访问LFS网站http://www.linuxfromscratch.org/mirrors.html 获取当前镜像列表。

1.4.5。联系信息

请将您的所有问题和评论定向到LFS邮件列表之一（请参见上文）。

1.5。救命

如果在阅读本书时遇到问题或疑问，请查看FAQ页面，网址为http://www.linuxfromscratch.org/faq/#generalfaq。那里的问题通常已经回答了。如果在此页面上未回答您的问题，请尝试查找问题的根源。以下提示将为您提供一些故障排除指南：http://www.linuxfromscratch.org/hints/downloads/files/errors.txt。

如果找不到在FAQ中列出的问题，请在http://www.linuxfromscratch.org/search.html上搜索邮件列表。

我们还有一个很棒的LFS社区，愿意通过邮件列表和IRC提供帮助（请参见本书的第1.4节“资源”部分）。但是，我们每天都会收到一些支持问题，通过访问FAQ并首先搜索邮件列表，可以很容易地回答许多问题。因此，为了使我们能够提供最佳的帮助，您需要首先自己进行一些研究。这使我们能够专注于更加特殊的支持需求。如果您的搜索没有找到解决方案，请在请求帮助中包括所有相关信息（如下所述）。

1.5.1。提及的事情

除了对所遇到问题的简要说明之外，任何请求帮助中都应包括的基本内容是：

- 正在使用的书籍的版本（在本例中为 9.0）
- 用于创建LFS的主机发行版和版本
- 主机系统需求脚本 的输出
- 在其中遇到问题的程序包或部分
- 收到确切的错误消息或症状
- 请注意您是否完全偏离了本书

> **注意**
>
> 这本书偏差也*并不*意味着我们不会帮助你。毕竟，LFS与个人喜好有关。预先确定已建立程序的任何更改，有助于我们评估和确定造成问题的可能原因。

1.5.2。配置脚本问题

如果在运行配置脚本时出现问题，请查看config.log文件。该文件可能包含在配置过程中遇到的未打印到屏幕上的错误。如果需要帮助，请包括*相关*行。

1.5.3。编译问题

屏幕输出和各种文件的内容都有助于确定编译问题的原因。配置脚本的屏幕输出和 make运行可能会有所帮助。不必包含整个输出，但必须包含足够的相关信息。以下是从make的屏幕输出中包含的信息类型的示例：

```
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
```

在这种情况下，很多人只需要包括下面的部分：

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

这是不足以正确诊断问题的信息，因为它仅指出出了什么问题而不是出了 *什么*问题。就像上面的示例一样，应该保存整个部分，因为它包括已执行的命令和相关的错误消息。

可以在**http://catb.org/~esr/faqs/smart-questions.html**在线获得有关在Internet上寻求帮助的出色文章。阅读并遵循本文档中的提示以增加获得所需帮助的可能性。

第二部分 为构建做准备

第2章准备主机系统

2.1。介绍

在本章中，将检查并安装构建LFS所需的主机工具。然后准备一个将托管LFS系统的分区。我们将创建分区本身，在分区上创建一个文件系统，然后挂载它。

2.2。主机系统要求

您的主机系统应具有以下软件，其中标出了最低版本。对于大多数现代Linux发行版来说，这应该不是问题。还要注意，许多发行版会将软件头文件放置在单独的软件包中，通常以“ <package-name> -devel ”或“ <package-name> -dev ”的形式。如果您的发行版提供它们，请确保安装它们。

列出的软件包的较早版本可能会运行，但尚未经过测试。

Bash-3.2（/ bin / sh应该是bash的符号链接或硬链接）

Binutils-2.25（不建议使用大于2.32的版本，因为它们尚未经过测试）

Bison-2.7（/ usr / bin / yacc应该是指向野牛或执行野牛的小脚本的链接）

Bzip2-1.0.4

Coreutils-6.9

Diffutils 2.8.1

Findutils-4.2.31

Gawk-4.0.1（/ usr / bin / awk应该是指向gawk的链接）

GCC-6.2， 包括C ++编译器，g ++（不建议使用高于9.2.0的版本，因为它们尚未经过测试）

Glibc-2.11（不建议使用高于2.30的版本，因为它们尚未经过测试）

Grep-2.5.1a

Gzip-1.3.12

Linux内核3.2

要求内核版本的原因是，在开发人员的建议下，我们在第6章中构建glibc时指定了该版本。udev也需要它。

如果主机内核早于3.2，则需要用更新的版本替换内核。有两种方法可以解决此问题。首先，查看您的Linux供应商是否提供3.2或更高版本的内核软件包。如果是这样，您可能希望安装它。如果您的供应商没有提供可接受的内核软件包，或者您不想安装它，则可以自己编译内核。第8章中提供了有关编译内核和配置引导加载程序（假设主机使用GRUB）的说明 。

M4-1.4.10

Make-4.0

补丁2.5.4

Perl-5.8.8

Python-3.4

Sed-4.1.5

焦油1.22

Texinfo-4.7

Xz-5.0.0

重要

请注意，使用本书中的说明来构建LFS系统需要上述符号链接。指向其他软件（例如破折号，Mawk等）的符号链接可能有效，但未经LFS开发团队测试或支持，并且可能需要偏离说明或某些软件包的附加补丁程序。

要查看您的主机系统是否具有所有适当的版本以及编译程序的能力，请运行以下命令：

```
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被设计为在一个会话中构建。也就是说，这些指令假定在此过程中不会关闭系统。这并不意味着该系统必须一次坐下来。问题是，如果在不同点恢复LFS，则某些过程必须在重新引导后重新完成。

2.3.1。第1-4章

这些章节在主机系统上完成。重新启动时，请注意以下事项：

- 在2.4节之后，以root用户身份完成的过程需要为*ROOT USER*设置LFS环境变量。

2.3.2。第五章

- 必须挂载/ mnt / lfs分区。
- 第5章中的*所有*指令必须由用户 *lfs*完成。在执行第5章中的任何任务之前，必须完成一个自检。

- 在该程序第5.3节，“一般编辑指令”是至关重要的。如果对安装软件包有任何疑问，请确保删除任何先前扩展的tarball，重新提取软件包文件，并完成该部分中的所有说明。

2.3.3。第6-8章

- 必须挂载/ mnt / lfs分区。
- 输入chroot时，必须为root设置LFS环境变量。否则，不使用LFS变量。
- 必须安装虚拟文件系统。可以在进入chroot之前或之后通过更改为主机虚拟终端并以root用户身份运行第6.2.2节“安装和填充/ dev”和 第6.2.3节“安装虚拟内核文件系统”中的命令来完成此操作。。

2.4。创建一个新分区

与大多数其他操作系统一样，LFS通常安装在专用分区上。建议的构建LFS系统的方法是使用可用的空分区，或者，如果您有足够的未分区空间，则创建一个空分区。

最小的系统需要大约6 GB的分区。这足以存储所有源tarball并编译软件包。但是，如果LFS系统打算用作主要的Linux系统，则可能会安装其他软件，这将需要额外的空间。20 GB分区是一个合理的大小，可以提供增长空间。LFS系统本身不会占用太多空间。此需求的很大一部分是提供足够的免费临时存储，以及在LFS完成后添加其他功能。此外，编译软件包可能需要大量磁盘空间，在安装软件包后，这些磁盘空间将被回收。

因为并非总是有足够的随机存取存储器（RAM）用于编译过程，所以最好使用小的磁盘分区作为*swap* 空间。内核使用它来存储很少使用的数据，并为活动进程保留更多的可用内存。*swap*LFS系统的分区可以与主机系统使用的 分区相同，在这种情况下，无需创建另一个分区。

使用命令行选项 启动磁盘分区程序（例如cfdisk或fdisk），命名将在其上创建新分区的硬盘（例如/dev/sda主磁盘驱动器）。创建Linux本机分区和一个*swap*分区（如果需要）。请参阅 cfdisk(8)或fdisk(8)如果您尚不知道如何使用程序。

> **注意**
>
> 对于有经验的用户，其他分区方案也是可能的。新的LFS系统可以在软件 **RAID**阵列或 **LVM**逻辑卷上。但是，其中一些选项需要使用 **initramfs**，这是一个高级主题。不建议首次使用LFS的用户使用这些分区方法。

记住新分区的名称（例如 sda5）。本书将其称为LFS分区。还请记住 *swap*分区的名称。这些名称将在以后用于/etc/fstab文件中。

2.4.1。其他分区问题

有关系统分区的建议请求通常发布在LFS邮件列表上。这是一个非常主观的话题。大多数发行版的默认设置是使用整个驱动器，但一个小型交换分区除外。由于以下几个原因，这对于LFS并不是最佳的。它降低了灵活性，使跨多个发行版的数据共享或LFS构建变得更加困难，使备份更加耗时，并且由于文件系统结构的低效率分配而浪费了磁盘空间。

2.4.1.1。根分区

/root十GB 的根LFS分区（不要与目录混淆 ）对于大多数系统来说是一个不错的选择。它提供了足够的空间来构建LFS和大多数BLFS，但又足够小，因此可以轻松创建多个分区进行实验。

2.4.1.2。交换分区

大多数发行版会自动创建交换分区。通常，交换分区的建议大小约为物理RAM大小的两倍，但这很少需要。如果磁盘空间有限，则将交换分区的容量保持在2 GB，并监视磁盘交换量。

交换永远都不是好事。通常，您可以通过侦听磁盘活动并观察系统对命令的反应来判断系统是否正在交换。交换的第一个反应应该是检查不合理的命令，例如尝试编辑5 GB的文件。如果交换是正常的情况，最好的解决方案是为系统购买更多的RAM。

2.4.1.3。Grub Bios分区

如果*引导磁盘*已使用GUID分区表（GPT）进行了分区，则必须创建一个小分区（通常为1 MB）（如果尚不存在）。该分区未格式化，但是在引导加载程序安装期间必须可供GRUB使用。如果使用fdisk，则该分区通常被标记为'BIOS Boot' ；如果使用gdisk，则 该分区的代码为 *EF02*。

> **注意**
>
> Grub Bios分区必须位于BIOS用于引导系统的驱动器上。这不一定是LFS根分区所在的驱动器。系统上的磁盘可能使用不同的分区表类型。此分区的要求仅取决于启动磁盘的分区表类型。

2.4.1.4。便利分区

还有其他一些分区不是必需的，但在设计磁盘布局时应考虑这些分区。以下列表并不全面，但仅作为参考。

- / boot –强烈建议。使用此分区来存储内核和其他引导信息。为了最大程度地减少较大磁盘的启动问题，请将其作为第一个磁盘驱动器上的第一个物理分区。100 MB的分区大小已足够。
- / home –强烈建议。在多个发行版或LFS构建中共享您的主目录和用户自定义。该大小通常很大，并且取决于可用的磁盘空间。
- / usr –如果为瘦客户机或无盘工作站提供服务器，则通常使用单独的/ usr分区。LFS通常不需要它。5 GB的大小可处理大多数安装。
- / opt –此目录对于BLFS最为有用，在BLFS中，可以安装大型软件包（如Gnome或KDE）的多个安装，而无需将文件嵌入/ usr层次结构中。如果使用5到10 GB，通常就足够了。
- / tmp –单独的/ tmp目录很少，但是在配置瘦客户机时很有用。如果使用此分区，通常将不需要超过几个千兆字节。
- / usr / src –该分区对于提供存储BLFS源文件并在LFS构建之间共享它们的位置非常有用。它也可以用作构建BLFS软件包的位置。30-50 GB的合理大分区可提供足够的空间。

需要在启动时自动挂载的任何单独分区都需要在中指定/etc/fstab。有关如何指定分区的详细信息将在第8.2节“创建/ etc / fstab文件”中进行讨论。

2.5。在分区上创建文件系统

现在已经建立了一个空白分区，可以创建文件系统了。LFS可以使用Linux内核认可的任何文件系统，但是最常见的类型是ext3和ext4。文件系统的选择可能很复杂，并且取决于文件的特征和分区的大小。例如：

ext2
适用于不经常更新的小型分区，例如/ boot。

ext3
是对ext2的升级，其中包括一个日志，可在不正常关机的情况下帮助恢复分区的状态。它通常用作通用文件系统。

ext4
是ext文件系统分区类型家族的最新版本。它提供了几项新功能，包括纳秒级时间戳，超大文件（16 TB）的创建和使用以及速度的提高。

其他文件系统，包括FAT32，NTFS，ReiserFS，JFS和XFS，对于特殊目的也很有用。有关这些文件系统的更多信息，请参见 **http://en.wikipedia.org/wiki/Comparison_of_file_systems**。

LFS假定根文件系统（/）的类型为ext4。要*ext4*在LFS分区上创建文件系统，请运行以下命令：

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

如果使用的是现有*swap*分区，则无需格式化它。如果*swap*创建了新分区，则需要使用以下命令对其进行初始化：

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

用 分区*<yyy>* 名称替换*swap*。

2.6。设置$ LFS变量

在本书中，环境变量LFS将被多次使用。您应该确保在整个LFS构建过程中始终定义此变量。应该将其设置为您将在其中构建LFS系统的目录的名称-我们将 /mnt/lfs作为示例，但是目录的选择取决于您。如果要在单独的分区上构建LFS，则此目录将是该分区的安装点。选择目录位置，然后使用以下命令设置变量：

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

设置此变量的好处在于，可以直接键入mkdir -v $ LFS / tools之类的命令 。壳将自动替换“ $ LFS ”与“ 的/ mnt / LFS ”（或任何变量设置为）当它处理命令行。

警告

不要忘记检查LFS是只要你离开，并设置重新输入当前的工作环境（例如做当为苏来*root*或其他用户）。使用以下命令检查 LFS变量设置是否正确：

```
echo $LFS
```

确保输出显示了您的LFS系统构建位置的路径，/mnt/lfs是否遵循提供的示例。如果输出不正确，请使用本页前面给出的命令将其设置$LFS 为正确的目录名称。

注意

确保LFS 始终设置变量的一种方法是.bash_profile在个人主目录中和两者中编辑文件，然后在/root/.bash_profile上方输入export命令。另外，在/etc/passwd文件中为需要该LFS变量的所有用户指定的外壳 程序需要进行bash操作，以确保将/root/.bash_profile文件合并为登录过程的一部分。

另一个考虑因素是用于登录主机系统的方法。如果通过图形显示管理器登录，则在.bash_profile启动虚拟终端时通常不会使用该用户。在这种情况下，请将export命令添加到.bashrc用户和root用户的文件中。此外，某些发行版还包含一些指令，这些指令不能.bashrc在非交互式bash调用中运行这些指令。确保在测试之前添加导出命令以用于非交互使用。

2.7。挂载新分区

现在已经创建了文件系统，需要使分区可访问。为此，需要将分区安装在选定的安装点。就本书而言，假定文件系统已安装LFS 在上一节中所述的环境变量指定的目录下。

通过运行以下命令创建安装点并安装LFS文件系统：

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

替换<xxx>为LFS分区的名称。

如果对LFS使用多个分区（例如，一个用于 /，另一个用于/usr），请使用以下命令安装它们：

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

用适当的分区名称替换<xxx>和<yyy>。

确保未使用过于严格的权限（例如nosuid或nodev 选项）装入此新分区。运行不带任何参数的mount命令，以查看为已安装的LFS分区设置了哪些选项。如果设置了nosuid和/或 nodev，则需要重新安装分区。

警告

以上说明假定您不会在整个LFS过程中重新启动计算机。如果关闭系统，则每次重新启动构建过程时都需要重新挂载LFS分区，或者修改主机系统的/etc / fstab文件以在引导时自动重新挂载它。例如：

```
/ dev / <xxx> / mnt / lfs ext4默认值1 1
```

如果使用其他可选分区，请确保也添加它们。

如果使用*swap* 分区，请确保使用swapon命令启用该分区：

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

用 分区*<zzz>* 名称替换*swap*。

现在已经有了一个可以工作的地方，是时候下载软件包了。

第3章包装和补丁

3.1。介绍

本章包含了要构建基本Linux系统需要下载的软件包的列表。列出的版本号对应于已知可以运行的软件版本，并且本书基于其使用情况。我们强烈建议您不要使用较新的版本，因为一个版本的构建命令可能不适用于较新的版本。最新的软件包版本也可能存在需要解决的问题。这些变通办法将在本书的开发版本中进行开发和稳定。

下载位置可能并不总是可以访问。如果自本书出版以来下载位置已更改，则Google（**http://www.google.com/**）为大多数软件包提供了有用的搜索引擎。如果搜索失败，请尝试在**http://www.linuxfromscratch.org/lfs/packages.html#packages中**讨论的另一种下载方法。

下载的软件包和修补程序将需要存储在整个构建中都方便使用的位置。还需要一个工作目录来解压缩源并构建它们。$LFS/sources既可以用作存储tarball和补丁的位置，又可以用作工作目录。通过使用此目录，所需的元素将位于LFS分区上，并且在构建过程的所有阶段都可用。

要创建此目录，请*root*在开始下载会话之前以用户身份执行以下命令：

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

使该目录可写且具有粘性。“ 粘滞 ”是指即使多个用户对目录具有写权限，也只有文件所有者才能删除粘滞目录中的文件。以下命令将启用写和粘滞模式：

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

下载所有软件包和补丁的一种简单方法是使用**wget-list**作为wget的输入。例如：

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

此外，从LFS-7.0开始，还有一个单独的文件 **md5sums**，可用于在继续操作之前验证所有正确的软件包是否可用。将该文件放入$LFS/sources并运行：

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

3.2。所有套餐

下载或以其他方式获得以下软件包：

- **Acl (2.2.53) -513 KB：**
  下载：**http**：**//download.savannah.gnu.org/releases/acl/acl-2.2.53.tar.gz**
  MD5总和： 007aabf1dbb550bcddde52a244cd1070
- **Attr (2.4.48) -457 KB：**
  主页：**https**：**//savannah.nongnu.org/projects/attr**
  下载：**http**：**//download.savannah.gnu.org/releases/attr/attr-2.4.48.tar.gz**
  MD5总和： bc1e5cb5c96d99b24886f1f527d3bb3d
- **Autoconf (2.69) -1,186 KB：**
  主页：**http**：**//www.gnu.org/software/autoconf/**
  下载：**http**：**//ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz**
  MD5总和： 50f97f4159805e374639a73e2636f22e
- **Automake (1.16.1) -1,499 KB：**
  主页：**http**：**//www.gnu.org/software/automake/**
  下载：**http**：**//ftp.gnu.org/gnu/automake/automake-1.16.1.tar.xz**
  MD5总和： 53f38e7591fa57c3d2cee682be668e5b
- **重击 (5.0) -9,898 KB：**
  主页：**http**：**//www.gnu.org/software/bash/**
  下载：**http**：**//ftp.gnu.org/gnu/bash/bash-5.0.tar.gz**
  MD5总和： 2b44b47b905be16f45709648f671820b
- **Bc (2.1.3) -233 KB：**
  主页：**https**：**//github.com/gavinhoward/bc**
  下载：**https**：**//github.com/gavinhoward/bc/archive/2.1.3/bc-2.1.3.tar.gz**
  MD5总和： 2a882dc39c0fb8e36c12f590d54cc039
- **Binutils (2.32) -20,288 KB：**
  主页：**http**：**//www.gnu.org/software/binutils/**

下载：http：//ftp.gnu.org/gnu/binutils/binutils-2.32.tar.xz
MD5总和： 0d174cdaf85721c5723bf52355be41e6

- **野牛 (3.4.1) -2,147 KB：**
  主页：http：//www.gnu.org/software/bison/
  下载：http：//ftp.gnu.org/gnu/bison/bison-3.4.1.tar.xz
  MD5总和： 201286a573b12da109df96282fe4ff4a
- **Bzip2 (1.0.8) -792 KB：**
  下载：https：//www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz
  MD5总和： 67e051268d0c475ea773822f7500d0e5
- **检查 (0.12.0) -747 KB：**
  主页：https：//libcheck.github.io/check
  下载：https：//github.com/libcheck/check/releases/download/0.12.0/check-0.12.0.tar.gz
  MD5总和： 31b17c6075820a434119592941186f70
- **Coreutils (8.31) -5,284 KB：**
  主页：http：//www.gnu.org/software/coreutils/
  下载：http：//ftp.gnu.org/gnu/coreutils/coreutils-8.31.tar.xz
  MD5总和： 0009a224d8e288e8ec406ef0161f9293
- **DejaGNU (1.6.2) -514 KB：**
  主页：http：//www.gnu.org/software/dejagnu/
  下载：http：//ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.2.tar.gz
  MD5总和： e1b07516533f351b3aba3423fafeffd6
- **Diffutils (3.7) -1,415 KB：**
  主页：http：//www.gnu.org/software/diffutils/
  下载：http：//ftp.gnu.org/gnu/diffutils/diffutils-3.7.tar.xz
  MD5总和： 4824adc0e95dbbf11dfbdfaad6a1e461
- **E2fsprogs (1.45.3) -7,741 KB：**
  主页：http：//e2fsprogs.sourceforge.net/
  下载：https：//downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.45.3/e2fsprogs-1.45.3.tar.gz
  MD5总和： 447a225c05f0a81121f6ddffbf55b06c
- **Elfutils (0.177) -8,645 KB：**
  主页：https：//sourceware.org/ftp/elfutils/
  下载：https：//sourceware.org/ftp/elfutils/0.177/elfutils-0.177.tar.bz2
  MD5总和： 0b583722f911e1632544718d502aab87
- **Eudev (3.2.8) -1,850 KB：**
  下载：https：//dev.gentoo.org/~blueness/eudev/eudev-3.2.8.tar.gz
  MD5总和： ce166b3fdd910c2a4a840378f48fedaf
- **外国人 (2.2.7) -415 KB：**
  主页：https：//libexpat.github.io/
  下载：https：//prdownloads.sourceforge.net/expat/expat-2.2.7.tar.xz

MD5总和： 3659bc0938db78815b5f5a9c24d732aa

- **预期 (5.45.4) -618 KB：**
  主页：https：//core.tcl.tk/expect/
  下载： https：//prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz
  MD5总和： 00fce8de158422f5ccd2666512329bd2

- **文件 (5.37) -867 KB：**
  主页：https：//www.darwinsys.com/file/
  下载：ftp：//ftp.astron.com/pub/file/file-5.37.tar.gz
  MD5总和： 80c29aca745466c6c24d11f059329075

> **注意**
>
> 文件 (5.37) 在所列位置可能不再可用。当新版本发布时，主下载位置的站点管理员有时会删除旧版本。也可以在以下位置找到可能具有正确版本的备用下载位置： http：//www.linuxfromscratch.org/lfs/download.html#ftp。

- **Findutils (4.6.0) -3,692 KB：**
  主页：http：//www.gnu.org/software/findutils/
  下载： http：//ftp.gnu.org/gnu/findutils/findutils-4.6.0.tar.gz
  MD5总和： 9936aa8009438ce185bea2694a997fc1

- **Flex (2.6.4) -1,386 KB：**
  主页：https：//github.com/westes/flex
  下载： https：//github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz
  MD5总和： 2882e3179748cc9f9c23ec593d6adc8d

- **Gawk (5.0.1) -3,063 KB：**
  主页：http：//www.gnu.org/software/gawk/
  下载：http：//ftp.gnu.org/gnu/gawk/gawk-5.0.1.tar.xz
  MD5总和： f9db3f6715207c6f13719713abc9c707

- **GCC (9.2.0) -68,953 KB：**
  主页：https：//gcc.gnu.org/
  下载： http：//ftp.gnu.org/gnu/gcc/gcc-9.2.0/gcc-9.2.0.tar.xz
  MD5总和： 3818ad8600447f05349098232c2ddc78

- **GDBM (1.18.1) -920 KB：**
  主页：http：//www.gnu.org/software/gdbm/
  下载：http：//ftp.gnu.org/gnu/gdbm/gdbm-1.18.1.tar.gz
  MD5总和： 988dc82182121c7570e0cb8b4fcd5415

- **Gettext (0.20.1) -9,128 KB：**
  主页：http：//www.gnu.org/software/gettext/
  下载：http：//ftp.gnu.org/gnu/gettext/gettext-0.20.1.tar.xz
  MD5总和： 9ed9e26ab613b668e0026222a9c23639

- **Glibc (2.30) -16,189 KB :**
  主页：http：//www.gnu.org/software/libc/
  下载：http：//ftp.gnu.org/gnu/glibc/glibc-2.30.tar.xz
  MD5总和： 2b1dbdf27b28620752956c061d62f60c

- **GMP (6.1.2) -1,901 KB :**
  主页：http：//www.gnu.org/software/gmp/
  下载：http：//ftp.gnu.org/gnu/gmp/gmp-6.1.2.tar.xz
  MD5总和： f58fa8001d60c4c77595fbbb62b63c1d

- **Gperf (3.1) -1,188 KB :**
  主页：http：//www.gnu.org/software/gperf/
  下载：http：//ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz
  MD5总和： 9e251c0a618ad0824b51117d5d9db87e

- **Grep (3.3) -1,440 KB :**
  主页：http：//www.gnu.org/software/grep/
  下载：http：//ftp.gnu.org/gnu/grep/grep-3.3.tar.xz
  MD5总和： 05d0718a1b7cc706a4bdf8115363f1ed

- **Groff (1.22.4) -4,044 KB :**
  主页：http：//www.gnu.org/software/groff/
  下载：http：//ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz
  MD5总和： 08fb04335e2f5e73f23ea4c3adbf0c5f

- **GRUB (2.04) -6,245 KB :**
  主页：http：//www.gnu.org/software/grub/
  下载：https：//ftp.gnu.org/gnu/grub/grub-2.04.tar.xz
  MD5总和： 5aaca6713b47ca2456d8324a58755ac7

- **Gzip (1.10) -757 KB :**
  主页：http：//www.gnu.org/software/gzip/
  下载：http：//ftp.gnu.org/gnu/gzip/gzip-1.10.tar.xz
  MD5总和： 691b1221694c3394f1c537df4eee39d3

- **Iana-Etc (2.30) -201 KB :**
  主页：http：//freecode.com/projects/iana-etc
  下载：http：//anduin.linuxfromscratch.org/LFS/iana-etc-2.30.tar.bz2
  MD5总和： 3ba3afb1d1b261383d247f46cb135ee8

- **Inetutils (1.9.4) -1,333 KB :**
  主页：http：//www.gnu.org/software/inetutils/
  下载：http：//ftp.gnu.org/gnu/inetutils/inetutils-1.9.4.tar.xz
  MD5总和： 87fef1fa3f603aef11c41dcc097af75e

- **Intltool (0.51.0) -159 KB :**
  主页：https：//freedesktop.org/wiki/Software/intltool
  下载：https：//launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz
  MD5总和： 12e517cac2b57a0121cda351570f1e63

- **IPRoute2 (5.2.0) -713 KB :**
  主页 : https : //www.kernel.org/pub/linux/utils/net/iproute2/
  下载 : https : //www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.2.0.tar.xz
  MD5总和 : 0cb2736e7bc2f56254a363d3d23703b7

- **Kbd (2.2.0) -1,090 KB :**
  主页 : http : //ftp.altlinux.org/pub/people/legion/kbd
  下载 : https : //www.kernel.org/pub/linux/utils/kbd/kbd-2.2.0.tar.xz
  MD5总和 : d1d7ae0b5fb875dc082731e09cd0c8bc

- **Kmod (26) -540 KB :**
  下载 : https : //www.kernel.org/pub/linux/utils/kernel/kmod/kmod-26.tar.xz
  MD5总和 : 1129c243199bdd7db01b55a61aa19601

- **较少 (551) -339 KB :**
  主页 : http : //www.greenwoodsoftware.com/less/
  下载 : http : //www.greenwoodsoftware.com/less/less-551.tar.gz
  MD5总和 : 4ad4408b06d7a6626a055cb453f36819

- **LFS-启动脚本 (20190524) -32 KB :**
  下载 : http : //www.linuxfromscratch.org/lfs/downloads/9.0/lfs-bootscripts-20190524.tar.xz
  MD5总和 : c91b11e366649c9cec60c2552820fed5

- **Libcap (2.27) -67 KB :**
  主页 : https : //sites.google.com/site/fullycapable/
  下载 : https : //www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.27.tar.xz
  MD5总和 : 2e8f9fab32eb5ccb37969fe317fd17aa

- **Libffi (3.2.1) -920 KB :**
  主页 : https : //sourceware.org/libffi/
  下载 : ftp : //sourceware.org/pub/libffi/libffi-3.2.1.tar.gz
  MD5总和 : 83b89587607e3eb65c70d361f13bab43

- **Libpipeline (1.5.1) -965 KB :**
  主页 : http : //libpipeline.nongnu.org/
  下载 : http : //download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.1.tar.gz
  MD5总和 : 4c8fe6cd85422baafd6e060f896c61bc

- **Libtool (2.4.6) -951 KB :**
  主页 : http : //www.gnu.org/software/libtool/
  下载 : http : //ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz
  MD5总和 : 1bfb9b923f2c1339b4d2ce1807064aa5

- **Linux (5.2.8) -104,555 KB :**
  主页 : https : //www.kernel.org/
  下载 : https : //www.kernel.org/pub/linux/kernel/v5.x/linux-5.2.8.tar.xz
  MD5总和 : 602dd0ecb8646e539fefb2beb6eb6fe0

> **注意**
>
> 由于发现了安全漏洞，Linux内核被相对频繁地更新了很多次。除非勘误页面另有说明，否则应使用最新的可用5.2.x内核版本。
> 对于希望更新Linux内核的速度有限或带宽昂贵的用户，可以分别下载软件包和补丁程序的基准版本。这可能会为次要版本中的后续修补程序级别升级节省一些时间或成本。

- **M4 (1.4.18) -1,180 KB：**
  主页：**http：//www.gnu.org/software/m4/**
  下载：**http：//ftp.gnu.org/gnu/m4/m4-1.4.18.tar.xz**
  MD5总和： `730bb15d96fffe47e148d1e09235af82`
- **使 (4.2.1) -1,932 KB：**
  主页：**http：//www.gnu.org/software/make/**
  下载：**http：//ftp.gnu.org/gnu/make/make-4.2.1.tar.gz**
  MD5总和： `7d0dcb6c474b258aab4d54098f2cf5a7`
- **人力数据库 (2.8.6.1) -1,787 KB：**
  主页：**https：//www.nongnu.org/man-db/**
  下载：**http：//download.savannah.gnu.org/releases/man-db/man-db-2.8.6.1.tar.xz**
  MD5总和： `22e82fe1127f4ca95de7100168a927d1`
- **手册页 (5.02) -1,630 KB：**
  主页：**https：//www.kernel.org/doc/man-pages/**
  下载：**https：//www.kernel.org/pub/linux/docs/man-pages/man-pages-5.02.tar.xz**
  MD5总和： `136e5e3380963571a079693d8ae38f52`
- **介子 (0.51.1) -1,418 KB：**
  主页：**https：//mesonbuild.com**
  下载：**https：//github.com/mesonbuild/meson/releases/download/0.51.1/meson-0.51.1.tar.gz**
  MD5总和： `48787e391ec5c052799a3dd491f73909`
- **MPC (1.1.0) -685 KB：**
  主页：**http：//www.multiprecision.org/**
  下载：**https：//ftp.gnu.org/gnu/mpc/mpc-1.1.0.tar.gz**
  MD5总和： `4125404e41e482ec68282a2e687f6c73`
- **MPFR (4.0.2) -1,409 KB：**
  主页：**https：//www.mpfr.org/**
  下载：**http：//www.mpfr.org/mpfr-4.0.2/mpfr-4.0.2.tar.xz**
  MD5总和： `320fbc4463d4c8cb1e566929d8adc4f8`
- **忍者 (1.9.0) -187 KB：**
  主页：**https：//ninja-build.org/**
  下载：**https：//github.com/ninja-build/ninja/archive/v1.9.0/ninja-1.9.0.tar.gz**
  MD5总和： `f340be768a76724b83e6daab69009902`
- **Ncurses (6.1) -3,288 KB：**

主页：http：//www.gnu.org/software/ncurses/
下载：http：//ftp.gnu.org/gnu/ncurses/ncurses-6.1.tar.gz
MD5总和： 98c889aaf8d23910d2b92d65be2e737a

- **OpenSSL (1.1.1c) -8,657 KB：**
  主页：https：//www.openssl.org/
  下载：https：//www.openssl.org/source/openssl-1.1.1c.tar.gz
  MD5总和： 15e21da6efe8aa0e0768ffd8cd37a5f6
- **补丁 (2.7.6) -766 KB：**
  主页：https：//savannah.gnu.org/projects/patch/
  下载：http：//ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz
  MD5总和： 78ad9937e4caadcba1526ef1853730d5
- **Perl (5.30.0) -12,129 KB：**
  主页：https：//www.perl.org/
  下载：https：//www.cpan.org/src/5.0/perl-5.30.0.tar.xz
  MD5总和： 037c35000550bdcb47552ad0f6d3064d
- **Pkg-config (0.29.2) -1,970 KB：**
  主页：https：//www.freedesktop.org/wiki/Software/pkg-config
  下载：https：//pkg-config.freedesktop.org/releases/pkg-config-0.29.2.tar.gz
  MD5总和： f6e931e319531b736fadc017f470e68a
- **Procps (3.3.15) -884 KB：**
  主页：https：//sourceforge.net/projects/procps-ng
  下载：https：//sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.15.tar.xz
  MD5总和： 2b0717a7cb474b3d6dfdeedfbad2eccc
- **伪 (23.2) -297 KB：**
  主页：http：//psmisc.sourceforge.net/
  下载：https：//sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.2.tar.xz
  MD5总和： 0524258861f00be1a02d27d39d8e5e62
- **Python (3.7.4) -16,730 KB：**
  主页：https：//www.python.org/
  下载：https：//www.python.org/ftp/python/3.7.4/Python-3.7.4.tar.xz
  MD5总和： d33e4aae66097051c2eca45ee3604803
- **Python文档 (3.7.4) -6,068 KB：**
  下载：https：//docs.python.org/ftp/python/doc/3.7.4/python-3.7.4-docs-html.tar.bz2
  MD5总和： c410337e954dbba2d04fe169c355a6a2
- **Readline (8.0) -2,907 KB：**
  主页：https：//tiswww.case.edu/php/chet/readline/rltop.html
  下载：http：//ftp.gnu.org/gnu/readline/readline-8.0.tar.gz
  MD5总和： 7e6c1f16aee3244a69aba6e438295ca3
- **桑德 (4.7) -1,268 KB：**
  主页：http：//www.gnu.org/software/sed/

下载：http：//ftp.gnu.org/gnu/sed/sed-4.7.tar.xz
MD5总和： 777ddfd9d71dd06711fe91f0925e1573

- **阴影 (4.7) -1,587 KB：**
  下载：https：//github.com/shadow-maint/shadow/releases/download/4.7/shadow-4.7.tar.xz
  MD5总和： f7ce18c8dfd05f1a009266cb604d58b7
- **Sysklogd (1.5.1) -88 KB：**
  主页：http：//www.infodrom.org/projects/sysklogd/
  下载：http：//www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.1.tar.gz
  MD5总和： c70599ab0d037fde724f7210c2c8d7f8
- **Sysvinit (2.95) -122 KB：**
  主页：https：//savannah.nongnu.org/projects/sysvinit
  下载：http：//download.savannah.gnu.org/releases/sysvinit/sysvinit-2.95.tar.xz
  MD5总和： 66f488c952c70ff4245774fc7e87e529
- **焦油 (1.32) -2,055 KB：**
  主页：http：//www.gnu.org/software/tar/
  下载：http：//ftp.gnu.org/gnu/tar/tar-1.32.tar.xz
  MD5总和： 83e38700a80a26e30b2df054e69956e5
- **Tcl (8.6.9) -9,772 KB：**
  主页：http：//tcl.sourceforge.net/
  下载：https：//downloads.sourceforge.net/tcl/tcl8.6.9-src.tar.gz
  MD5总和： aa0a121d95a0e7b73a036f26028538d4
- **Texinfo (6.6) -4,831 KB：**
  主页：http：//www.gnu.org/software/texinfo/
  下载：http：//ftp.gnu.org/gnu/texinfo/texinfo-6.6.tar.xz
  MD5总和： 5231da3e6aa106cd0532b8609e5b3702
- **时区数据 (2019b) -376 KB：**
  主页：https：//www.iana.org/time-zones
  下载：https：//www.iana.org/time-zones/repository/releases/tzdata2019b.tar.gz
  MD5总和： b26b5d7d844cb96c73ed2fb6d588daaf
- **Udev-lfs Tarball (udev-lfs-20171102) -11 KB：**
  下载：http：//anduin.linuxfromscratch.org/LFS/udev-lfs-20171102.tar.xz
  MD5总和： 27cd82f9a61422e186b9d6759ddf1634
- **Util-linux (2.34) -4,859 KB：**
  主页：http：//freecode.com/projects/util-linux
  下载：https：//www.kernel.org/pub/linux/utils/util-linux/v2.34/util-linux-2.34.tar.xz
  MD5总和： a78cbeaed9c39094b96a48ba8f891d50
- **Vim (8.1.1846) -14,078 KB：**
  主页：https：//www.vim.org
  下载：https：//github.com/vim/vim/archive/v8.1.1846/vim-8.1.1846.tar.gz

MD5总和： 4f129a05254d93c739fcede843df87df

- **XML ::解析器 (2.44) -232 KB：**
  主页：https：//github.com/chorny/XML-Parser
  下载： https：//cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.44.tar.gz
  MD5总和： af4813fe3952362451201ced6fbce379
- **Xz Utils (5.2.4) -1030 KB：**
  主页：https：//tukaani.org/xz
  下载：https：//tukaani.org/xz/xz-5.2.4.tar.xz
  MD5总和： 003e4d0b1b1899fc6e3000b24feddf7c
- **Zlib (1.2.11) -457 KB：**
  主页：https：//www.zlib.net/
  下载：https：//zlib.net/zlib-1.2.11.tar.xz
  MD5总和： 85adef240c5f370b308da8c938951a68

这些软件包的总大小：约391 MB

3.3。所需补丁

除软件包外，还需要几个补丁。这些补丁纠正了维护人员应修复的软件包中的任何错误。补丁程序还做了一些小的修改，以使软件包更易于使用。构建LFS系统将需要以下补丁：

- **Bzip2文档补丁 -1.6 KB：**
  下载： http：//www.linuxfromscratch.org/patches/lfs/9.0/bzip2-1.0.8-install_docs-1.patch
  MD5总和： 6a5ac7e89b791aae556de0f745916f7f
- **Coreutils国际化修复补丁-168 KB：**
  下载： http：//www.linuxfromscratch.org/patches/lfs/9.0/coreutils-8.31-i18n-1.patch
  MD5总和： a9404fb575dfd5514f3c8f4120f9ca7d
- **Glibc FHS补丁-2.8 KB：**
  下载： http：//www.linuxfromscratch.org/patches/lfs/9.0/glibc-2.30-fhs-1.patch
  MD5总和： 9a5997c3452909b1769918c759eff8a2
- **Kbd Backspace /删除修复补丁 -12 KB：**
  下载： http：//www.linuxfromscratch.org/patches/lfs/9.0/kbd-2.2.0-backspace-1.patch
  MD5总和： f75cca16a38da6caa7d52151f7136895
- **Sysvinit合并补丁 -2.4 KB：**
  下载： http：//www.linuxfromscratch.org/patches/lfs/9.0/sysvinit-2.95-consolidated-1.patch
  MD5总和： 4900322141d493e74020c9cf437b2cdc

这些补丁的总大小：约186.8 KB

除了上述必需的修补程序之外，LFS社区还创建了许多可选的修补程序。这些可选补丁解决了一些小问题或启用了默认情况下未启用的功能。可以随意浏览位于 **http://www.linuxfromscratch.org/patches/downloads/**的补丁程序数据库， 并获取任何其他适合您系统需求的补丁程序。

第4章最后准备

4.1。介绍

在本章中，我们将执行一些其他任务以准备构建临时系统。我们将在其中创建$LFS用于安装临时工具的目录，添加非特权用户以降低风险，并为该用户创建适当的构建环境。我们还将说明用于度量LFS软件包构建所需时间（或“ SBU ”）的时间单位，并提供有关软件包测试套件的一些信息。

4.2。创建$ LFS / tools目录

将把第5章中 编译的所有程序$LFS/tools与第6章中编译的程序分开安装。此处编译的程序是临时工具，不会成为最终LFS系统的一部分。通过将这些程序保存在单独的目录中，可以在使用后轻松将其丢弃。这也可以防止这些程序最终出现在宿主生产目录中（在第5章中，很容易偶然发生 ）。

通过运行以下命令来创建所需的目录 *root*：

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

下一步是/tools在主机系统上创建符号链接。这将指向LFS分区上新创建的目录。同样运行以下命令*root*：

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

> **注意**
>
> 上面的命令是正确的。该LN命令有一些语法变化，所以一定要检查信息coreutils的LN和 ln(1)报告您可能会认为是错误之前。

创建的symlink使工具链能够被编译，以便它始终引用/tools，这意味着编译器，汇编器和链接器将在第5章（当我们仍在使用主机中的某些工具时）和下一章（当我们在主机上）工作时被“ chroot ”到LFS分区）。

4.3。添加LFS用户

以用户身份登录时*root*，犯一个错误可能会损坏或破坏系统。因此，我们建议在本章中以非特权用户身份来构建软件包。您可以使用自己的用户名，但是要使其更易于设置干净的工作环境，请创建一个称为*lfs*新组成员（也称为*lfs*）的新用户，并在安装过程中使用该用户。作为*root*，发出以下命令来添加新用户：

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

命令行选项的含义：

-s /bin/bash
这使bash成为user的默认外壳*lfs*。

-g lfs
此选项将用户添加*lfs* 到组*lfs*。

-m
这将为创建一个主目录*lfs*。

-k /dev/null
/etc/skel通过将输入位置更改为特殊的空设备， 此参数可防止从框架目录（默认值为）复制文件。

lfs
这是创建的组和用户的实际名称。

要以身份登录*lfs*（而不*lfs*是以身份登录时切换到用户*root*，这不需要*lfs*用户输入密码），请*lfs*输入密码：

```
passwd lfs
```

通过使目录拥有者 授予*lfs*完全访问权限 ： $LFS/tools*lfs*

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

如果根据建议创建了一个单独的工作目录，请为用户授予*lfs*该目录的所有权：

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

接下来，以用户身份登录*lfs*。可以通过虚拟控制台，显示管理器或以下替代用户命令来完成此操作：

```
su - lfs
```

在“ -”指示 苏开始登录shell，而不是一个非登录shell。可以在bash(1)和info bash中找到这两种类型的shell之间的区别 。

4.4。搭建环境

通过为bash shell 创建两个新的启动文件来建立良好的工作环境。以用户身份登录时*lfs*，发出以下命令来创建新的.bash_profile：

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

当以用户身份登录时*lfs*，初始外壳通常是一个*登录*外壳，它将读取/etc/profile主机的（可能包含一些设置和环境变量），然后读取 .bash_profile。该 高管ENV -i ... /斌/bash中的命令.bash_profile文件替换为新的具有完全空的环境下运行的外壳，除了HOME，TERM和PS1 变量。这样可以确保主机系统不会有有害的环境变量，不会泄漏到构建环境中。这里使用的技术可以达到确保环境清洁的目的。

外壳程序的新实例*是非登录*外壳程序，它不读取/etc/profile或.bash_profile文件，而是读取 .bashrc文件。立即创建 .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
```

所述集+ H 命令关闭的bash的散列函数。哈希通常是一个有用的功能-bash使用哈希表来记住可执行文件的完整路径，以避免PATH一次又一次地查找同一可执行文件。但是，新工具应在安装后立即使用。通过关闭哈希功能，shell将始终PATH 在运行程序时进行搜索。这样，shell将在以下位置找到新编译的工具$LFS/tools 一旦可用，而无需在不同位置记住同一程序的先前版本。

将用户文件创建掩码（umask）设置为022，可以确保新创建的文件和目录只能由其所有者写，但任何人都可以读取和执行（假设open(2)系统调用使用默认模式，则新文件最终会带有权限模式644和目录模式755）。

该LFS变量应设置为所选的安装点。

该LC_ALL变量控制某些程序的本地化，使它们的消息遵循指定国家/地区的约定。设置LC_ALL为“ POSIX ”或“ C ”（两者等效）可确保一切在chroot环境中都能正常工作。

该LFS_TGT变量设置一个非默认但兼容的机器描述，以供构建我们的交叉编译器和链接器以及交叉编译临时工具链时使用。更多信息包含在 第5.2节“工具链技术说明”中。

通过/tools/bin领先于该标准PATH，安装 完第5章中安装的所有程序后，它们就会立即被外壳程序拾取。与第5章环境中可用的旧程序结合使用，可以关闭散列，从而限制了主机使用旧程序的风险。

最后，要为准备临时工具的环境做好充分准备，请提供刚刚创建的用户配置文件：

```
source ~/.bash_profile
```

4.5。关于SBU

许多人想事先知道编译和安装每个软件包需要多长时间。由于Linux From Scratch可以构建在许多不同的系统上，因此无法提供准确的时间估计。最大的软件包（Glibc）在最快的系统上大约需要20分钟，而在较慢的系统上最多可能需要三天！代替提供实际时间，将使用标准构建单位（SBU）度量。

SBU措施的工作原理如下。本书第一个要编译的软件包是第5章中的 Binutils 。编译此程序包所花费的时间就是所谓的标准构建单元或SBU。所有其他编译时间将相对于此时间表示。

例如，考虑一个编译时间为4.5 SBU的软件包。这意味着，如果系统花费10分钟来编译和安装Binutils的第一遍，那么构建此示例软件包将花费 *大约* 45分钟。幸运的是，大多数构建时间比Binutils的构建时间短。

通常，SBU并非完全准确，因为它们取决于许多因素，包括主机系统的GCC版本。这里提供它们是为了估计安装软件包可能需要多长时间，但是在某些情况下，数目可能会相差数十分钟。

注意

对于具有多个处理器（或内核）的许多现代系统，可以通过设置环境变量或告知make程序有多少个处理器来执行“并行make”来减少程序包的编译时间。例如，Core2Duo可以同时支持两个进程：

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

或仅使用

```
make -j2
```

当以这种方式使用多个处理器时，书中的SBU单元的变化将比平时更大。在某些情况下，make步骤将完全失败。分析构建过程的输出也将更加困难，因为不同过程的行将被交错。如果您在构建步骤中遇到问题，请恢复到单个处理器的构建以正确分析错误消息。

4.6。关于测试套件

大多数软件包都提供测试套件。为新建软件包运行测试套件是一个好主意，因为它可以提供“健全性检查”，表明一切都已正确编译。通过其一组检查的测试套件通常会证明该软件包可以按照开发人员的预期运行。但是，它不能保证该程序包完全没有错误。

一些测试套件比其他套件更重要。例如，核心工具链程序包（GCC，Binutils和Glibc）的测试套件由于在功能正常的系统中起着核心作用，因此至关重要。用于GCC和Glibc的测试套件可能需要很长时间才能完成，尤其是在速度较慢的硬件上，但是强烈建议您这样做。

注意

经验表明，运行第5章中的测试套件几乎无济于事。不可避免的事实是，主机系统总是对该章中的测试施加某些影响，通常会导致莫名其妙的故障。由于第5章中构建的工具 是临时工具 ，最终会被丢弃，因此我们不建议 普通读者运行第5章中的测试套件。提供了用于运行这些测试套件的说明，以使测试人员和开发人员受益，但这些说明严格是可选的。

运行Binutils和GCC测试套件的常见问题是伪终端（PTY）不足。这会导致大量失败的测试。发生这种情况可能有多种原因，但是最可能的原因是主机系统没有*devpts*正确设置文件系统。在**http://www.linuxfromscratch.org/lfs/faq.html#no-ptys上**将更详细地讨论此问题 。

有时，软件包测试套件会失败，但是由于开发人员意识到并且认为不重要的原因。请查阅位于**http://www.linuxfromscratch.org/lfs/build-logs/9.0/**的日志， 以验证是否会发生这些故障。该网站适用于本书中的所有测试。

第5章构建临时系统

5.1。介绍

本章介绍如何构建最小的Linux系统。该系统将仅包含足够的工具来开始在第6章中构建最终的LFS系统， 并为工作环境提供比最低环境更大的用户便利性。

构建此最小系统需要两个步骤。第一步是构建一个新的且独立于主机的工具链（编译器，汇编器，链接器，库和一些有用的实用程序）。第二步使用此工具链构建其他基本工具。

本章中编译的文件将安装在该`$LFS/tools`目录下， 以使其与下一章和主机生产目录中安装的文件分开。由于此处编译的软件包是临时的，因此我们不希望它们污染即将推出的LFS系统。

5.2。工具链技术说明

本节说明了整个构建方法背后的一些原理和技术细节。不必立即了解本节中的所有内容。在执行实际构建后，大多数信息将变得更加清晰。在此过程中的任何时候都可以引用此部分。

第5章 的总体目标 是创建一个临时区域，其中包含可以与主机系统隔离的一组已知良好的工具。通过使用 `chroot`，其余各章中的命令将包含在该环境中，从而确保目标LFS系统的构建干净无误。构建过程旨在将新读者的风险降至最低，并同时提供最大的教育价值。

注意

在继续之前，请注意工作平台的名称，通常称为目标三元组。确定目标三元组名称的一种简单方法是运行 许多软件包的源附带的 config.guess脚本。解压缩Binutils源文件并运行脚本：./config.guess并注意输出。例如，对于32位Intel处理器，输出将为*i686-pc-linux-gnu*。在64位系统上，它将为*x86_64-pc-linux-gnu*。

还请注意平台的动态链接程序的名称，通常称为动态加载程序（不要与Binutils 的标准链接程序ld混淆）。Glibc提供的动态链接器查找并加载程序所需的共享库，准备要运行的程序，然后运行它。对于32位Intel计算机，动态链接程序的名称为ld-linux.so.2（ld-linux-x86-64.so.2对于64位系统）。确定动态链接器名称的一种可靠方法是通过运行以下命令从主机系统检查随机二进制文件： readelf -l <name of binary> | grep interpreter并注意输出。涵盖所有平台的权威参考资料shlib-versions位于Glibc源代码树根目录中的文件中。

第5章 构建方法 如何工作的一些关键技术要点：

- 通过使用LFS_TGT变量来更改"供应商"字段目标三元组，来稍微调整工作平台的名称，以 确保Binutils和GCC的第一个版本产生兼容的交叉链接器和交叉编译器。交叉链接器和交叉编译器将生成与当前硬件兼容的二进制文件，而不是为其他体系结构生成二进制文件。
- 临时库是交叉编译的。由于交叉编译器本质上不能依赖于其宿主系统中的任何内容，因此该方法通过减少来自宿主的标头或库合并到新工具中的机会，消除了对目标系统的潜在污染。交叉编译还允许在支持64位的硬件上同时构建32位和64位库。
- 仔细操作GCC源会告诉编译器将使用哪个目标动态链接器。

首先安装Binutils是因为GCC和Glibc 的配置运行都会在汇编器和链接器上执行各种功能测试，以确定要启用或禁用的软件功能。这比人们可能首先意识到的更为重要。错误配置的GCC或Glibc可能会导致工具链微妙地断裂，这种断裂的影响可能要等到整个发行版本的构建接近尾声时才会显现出来。在执行过多的额外工作之前，测试套件故障通常会突出显示此错误。

Binutils的安装在两个位置的汇编器和连接器， /tools/bin以及/tools/$LFS_TGT/bin。一个位置的工具与另一个位置硬链接。链接器的重要方面是它的库搜索顺序。可以通过向ld传递 *--verbose*标志来获取详细信息。例如，ld --verbose | grep SEARCH将显示当前搜索路径及其顺序。通过编译一个伪程序并将开关传递给链接器，它显示ld链接了哪些文件 *--verbose*。例如，gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded将显示链接期间成功打开的所有文件。

安装的下一个软件包是GCC。在配置运行期间可以看到的示例是：

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

由于上述原因，这一点很重要。它还说明GCC的configure脚本不会搜索PATH目录来查找要使用的工具。但是，在gcc本身的实际操作过程中，不必使用相同的搜索路径。要找出gcc将使用哪个标准链接器，请运行：gcc -print-prog-name=ld。

通过在编译虚拟程序时向其传递 命令行选项， 可以从gcc获得详细信息*-v*。例如，gcc -v dummy.c将显示有关预处理器，编译和汇编阶段的详细信息，包括gcc包含的搜索路径及其顺序。

接下来安装的是经过清理的Linux API标头。这些允许标准C库（Glibc）与Linux内核将提供的功能进行接口。

下一个安装的软件包是Glibc。构建Glibc的最重要考虑因素是编译器，二进制工具和内核头文件。编译器通常不是问题，因为Glibc将始终使用与*--host*传递给配置脚本的参数有关的编译器 。例如，在我们的情况下，编译器将为 i686-lfs-linux-gnu-gcc。二进制工具和内核头文件可能更复杂。因此，不要冒险，并使用可用的配置开关强制执行正确的选择。运行 configure之后，检查config.make 文件中的内容。 glibc-build目录中的所有重要详细信息。请注意使用*CC="i686-lfs-gnu-gcc"*来控制使用哪些二进制工

具，使用*-nostdinc*和*-isystem*标志来控制编译器的包含搜索路径。这些项目突出显示了Glibc软件包的重要方面-它在构建机制方面非常自给自足，通常不依赖于工具链的默认设置。

在Binutils的第二遍过程中，我们能够使用 *--with-lib-path* configure开关来控制ld的库搜索路径。

对于GCC的第二遍，还需要修改其来源以告知GCC使用新的动态链接器。否则，将导致GCC程序本身具有/lib嵌入主机系统目录中的动态链接程序的名称，这将使脱离主机的目标无法实现。从这一点开始，核心工具链是独立的和托管的。第5章的其余 软件包均是针对于的新Glibc构建的/tools。

在第6章中进入chroot环境后，由于上面提到的自给性，第一个要安装的主要软件包是Glibc。一旦将此Glibc安装到中/usr，我们将对工具链默认值进行快速转换，然后继续构建目标LFS系统的其余部分。

5.3。通用编译说明

在构建软件包时，说明中有几个假设：

- 其中一些软件包在编译之前已进行了修补，但仅在需要修补才能解决问题时才进行修补。在本章和下一章中通常都需要一个补丁，但有时仅需要一个补丁。因此，不必担心是否缺少下载补丁的说明。应用补丁程序时，也可能会遇到有关*偏移*或 *模糊的*警告消息。不必担心这些警告，因为该补丁仍已成功应用。
- 在编译大多数软件包的过程中，屏幕上会滚动显示几个警告。这些是正常现象，可以放心忽略。这些警告如它们所显示的那样一警告已过时（但不是无效）使用C或C ++语法。C标准经常更改，并且某些软件包仍使用旧标准。这不是问题，但是会提示警告。
- 最后检查一次LFS是否正确设置了环境变量：

```
echo $LFS
```

/mnt/lfs使用我们的示例， 确保输出显示LFS分区的挂载点的路径。

- 最后，必须强调两个重要项目：

> **重要**
>
> 构建说明假定已正确设置了主机系统要求（包括符号链接）：
>
> - bash是使用中的外壳。
> - sh 是 bash的符号链接。
> - / usr / bin / awk是指向 gawk的符号链接。
> - / usr / bin / yacc 是野牛或执行野牛的小型脚本的符号链接。

重要

要重新强调构建过程：

1. 将所有源代码和修补程序放在可从chroot环境访问的目录中，例如/mnt/lfs/sources/。千万 ***不能***把来源/mnt/lfs/tools/。
2. 转到源目录。
3. 对于每个包装：
   a. 使用tar 程序，提取要构建的软件包。在第5章中，请确保您 在解压缩软件包时是 ***lfs***用户。
   b. 切换到提取软件包时创建的目录。
   c. 按照书中的说明构建软件包。
   d. 更改回源目录。
   e. 除非另有说明，否则删除提取的源目录。

5.4。Binutils-2.32-通过1

Binutils程序包包含一个链接器，一个汇编器和其他用于处理目标文件的工具。

	预计编制
时间：	1 SBU
所需磁盘空间：	580 MB

5.4.1。Cross Binutils的安装

注意

返回并重新阅读上一节中的注释。了解标注为重要的注释将在以后为您节省很多麻烦。

重要的是Binutils是第一个编译的软件包，因为Glibc和GCC都对可用的链接器和汇编器执行各种测试，以确定要启用其自身的功能。

Binutils文档建议在专用的构建目录中构建Binutils：

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

注意

为了使本书其余部分中列出的SBU值有用，请测量从配置到首次安装（包括首次安装）构建此软件包所花费的时间。容易地实现这一点，包装在命令 `时间` 这样的命令：`time { ./configure ... && ... && make install; }`。

注意

第5章中的大概构建SBU值和所需的磁盘空间不包括测试套件数据。

现在准备Binutils进行编译：

```
../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
这告诉配置脚本准备在`/tools`目录中安装Binutils程序。

--with-sysroot=$LFS
对于交叉编译，这告诉构建系统根据需要在$ LFS中查找目标系统库。

--with-lib-path=/tools/lib
这指定链接器应配置为使用哪个库路径。

`--target=$LFS_TGT`
由于`LFS_TGT`变量中的计算机描述与`config.guess`脚本返回的值略有不同，因此此开关将告诉`configure`脚本调整Binutil的构建系统以构建交叉链接器。

--disable-nls
这将禁用国际化，因为临时工具不需要i18n。

--disable-werror
这样可以防止在主机的编译器发出警告的情况下停止构建。

继续编译软件包：

```
make
```

编译完成。通常，我们现在将运行测试套件，但是在此早期阶段，测试套件框架（Tcl，Expect和DejaGNU）尚未到位。此时运行测试的好处是微不足道的，因为第一遍的程序很快就会被第二遍的程序所替代。

如果在x86_64上构建，请创建一个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
```

有关此软件包的详细信息，请参见第6.16.2节“ Binutils的内容”。

5.5。GCC-9.2.0-通过1

GCC软件包包含GNU编译器集合，其中包括C和C ++编译器。

	预计编制
时间：	12 SBU
所需磁盘空间：	3.1 GB

5.5.1。跨GCC的安装

GCC现在需要GMP，MPFR和MPC软件包。由于这些软件包可能未包含在您的主机发行版中，因此将使用GCC构建它们。将每个软件包解压缩到GCC源目录中，并重命名结果目录，以便GCC构建过程将自动使用它们：

> **注意**
>
> 关于本章经常有误会。这些步骤与前面所述的其他每一章相同（软件包构建说明）。首先从源目录中提取gcc tarball，然后转到创建的目录。只有这样，您才可以按照以下说明进行操作。

```
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
```

以下命令将更改GCC的默认动态链接器的位置，以使用安装在中的链接器 /tools。它还/usr/include从GCC的包含搜索路径中删除 。问题：

```
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
```

如果上述情况似乎难以理解，让我们对其进行分解。首先，我们将文件复制gcc/config/linux.h，gcc/config/i386/linux.h和gcc/config/i368/linux64.h同名的文件，但有一个附加的后缀“ .orig这样 ”。然后，第一sed的表达预规划“ /工具 ”到的每一个实例 “ / LIB / LD ”，“ / lib64下/ LD ”或“ / LIB32 / LD ”，而第二个取代的硬编码的实例“ 的/ usr ”。接下来，我们添加我们的define语句，这些语句将默认的startfile前缀更改为文件末尾。请注意，尾随“ / ”中的“ / lib中 / /工具 ”是必需的。最后，我们使用touch更新复制文件的时间戳。与cp -u结合使用时 ，可以防止在无意中运行两次命令时对原始文件进行意外更改。

最后，在x86_64主机上，将64位库的默认目录名称设置为“ lib ”：

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

GCC文档建议在专用的构建目录中构建GCC：

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

准备要编译的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-libquadmath                         \
    --disable-libssp                              \
    --disable-libvtv                              \
    --disable-libstdcxx                           \
    --enable-languages=c,c++
```

配置选项的含义：

--with-newlib

由于尚无可用的C库，因此这可以确保在构建libgcc时定义了hibit_libc常量。这样可以防止需要libc支持的任何代码的编译。

--without-headers

创建完整的交叉编译器时，GCC需要与目标系统兼容的标准头文件。为了我们的目的，将不需要这些标头。此开关可防止GCC寻找它们。

--with-local-prefix=/tools

本地前缀是GCC在系统中搜索本地安装的包含文件的位置。默认值为/usr/local。将此设置为/tools 有助于将主机位置保留在/usr/local此GCC的搜索路径之外。

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

默认情况下，GCC搜索/usr/include系统标头。结合sysroot开关，通常可以转换为$LFS/usr/include。但是，将在接下来的两个部分中安装的标头将转到$LFS/tools/include。此开关确保gcc可以正确找到它们。在GCC的第二遍中，同一开关将确保找不到主机系统中的标头。

--disable-shared

此开关强制GCC静态链接其内部库。我们这样做是为了避免主机系统可能出现的问题。

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

这些开关分别禁用对十进制浮点扩展，线程，libatomic，libgomp，libquadmath，libssp，libvtv和C ++标准库的支持。这些功能在构建交叉编译器时将无法编译，并且对于交叉编译临时libc的任务不是必需的。

--disable-multilib

在x86_64上，LFS尚不支持multilib配置。对于x86，此开关无害。

--enable-languages=c,c++

此选项可确保仅构建C和C ++编译器。这些是现在唯一需要的语言。

通过运行以下命令来编译GCC：

```
make
```

编译完成。此时，测试套件通常可以运行，但是，如前所述，测试套件框架尚未就绪。此时运行测试的好处是微不足道的，因为第一遍的程序将很快被替换。

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.21.2节“ GCC的内容”。

5.6。Linux-5.2.8 API标头

Linux API标头（在linux-5.2.8.tar.xz中）公开了内核的API，供Glibc使用。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	960 MB

5.6.1。安装Linux API标头

Linux内核需要公开一个应用程序编程接口（API），供系统的C库（LFS中的Glibc）使用。这是通过清理Linux内核源tarball中附带的各种C头文件来完成的。

确保软件包中没有嵌入过时的文件：

```
make mrproper
```

现在从源代码中提取用户可见的内核头文件。它们被放置在中间本地目录中并复制到所需的位置，因为提取过程会删除目标目录中的所有现有文件。

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

有关此软件包的详细信息，请参见第6.7.2节“ Linux API标头的内容”。

5.7。Glibc-2.30

Glibc软件包包含主要的C库。该库提供了用于分配内存，搜索目录，打开和关闭文件，读取和写入文件，字符串处理，模式匹配，算术等等的基本例程。

	预计编制
时间：	4.8 SBU
所需磁盘空间：	896 MB

5.7.1。安装Glibc

Glibc文档建议在专用的构建目录中构建Glibc：

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

接下来，准备编译Glibc：

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

配置选项的含义：

--host=$LFS_TGT, --build=$(../scripts/config.guess)
这些开关的综合作用是，Glibc的构建系统使用中的交叉链接器和交叉编译器将自身配置为交叉编译/tools。

--enable-kernel=3.2
这告诉Glibc编译支持3.2及更高版本Linux内核的库。未启用较早内核的解决方法。

--with-headers=/tools/include
这告诉Glibc可以根据最近安装到tools目录中的标头编译自己，以便它确切地知道内核具有什么功能，并可以相应地进行优化。

在此阶段，可能会出现以下警告：

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

msgfmt程序 丢失或不兼容通常是无害的。此msgfmt程序是主机发行版应提供的Gettext包的一部分。

> **注意**
>
> 有报道说，当以“并行方式”构建时，此软件包可能会失败。如果发生这种情况，请使用“ -j1”选项重新运行make命令。

编译软件包：

```
make
```

安装软件包：

```
make install
```

警告

在这一点上，必须停止并确保新工具链的基本功能（编译和链接）按预期工作。要执行健全性检查，请运行以下命令：

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

如果一切正常，则应该没有错误，并且最后一条命令的输出将采用以下格式：

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

请注意，对于32位计算机，解释器名称为 `/tools/lib/ld-linux.so.2`。

如果输出未如上所示或根本没有输出，则说明有问题。调查并追溯步骤以找出问题所在并纠正。必须继续解决此问题。

一旦一切顺利，请清理测试文件：

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

注意

接下来的部分中构建Binutils将额外检查工具链是否已正确构建。如果Binutils无法构建，则表明先前的Binutils，GCC或Glibc安装出现了问题。

有关此软件包的详细信息，请参见第6.9.3节“ Glibc的内容”。

5.8。来自GCC-9.2.0的Libstdc ++

Libstdc ++是标准的C ++库。需要编译C ++代码（GCC的一部分是用C ++编写的），但是在构建gcc-pass1时我们不得不推迟其安装，因为它依赖于glibc，而glibc在/ tools中尚不可用。

	预计编制
时间：	0.5 SBU
所需磁盘空间：	879 MB

5.8.1。安装目标Libstdc ++

> **注意**
>
> Libstdc ++是GCC来源的一部分。您应该首先解压缩GCC压缩包，然后转到gcc-9.2.0 目录。

为Libstdc ++创建一个单独的构建目录并输入：

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

准备Libstdc ++进行编译：

```
../libstdc++-v3/configure           \
    --host=$LFS_TGT                 \
    --prefix=/tools                 \
    --disable-multilib              \
    --disable-nls                   \
    --disable-libstdcxx-threads     \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/9.2.0
```

配置选项的含义：

--host=...
表示使用我们刚刚构建的交叉编译器，而不是中的一个/usr/bin。

--disable-libstdcxx-threads
由于我们尚未构建C线程库，因此也无法构建C ++。

--disable-libstdcxx-pch
此开关可防止安装当前阶段不需要的预编译包含文件。

--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/9.2.0

这是C ++编译器在其中搜索标准包含文件的位置。在正常构建中，此信息会自动从顶层目录传递到Libstdc ++ `configure`选项。在我们的情况下，必须明确提供此信息。

通过运行以下命令编译libstdc ++：

```
make
```

安装库：

```
make install
```

有关此软件包的详细信息，请参见第6.21.2节“ GCC的内容”。

5.9。Binutils-2.32-通行证2

Binutils程序包包含一个链接器，一个汇编器和其他用于处理目标文件的工具。

	预计编制
时间：	1.1 SBU
所需磁盘空间：	879 MB

5.9.1。安装Binutils

再次创建一个单独的构建目录：

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

准备Binutils进行编译：

```
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
```

新的配置选项的含义：

CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib
因为这实际上是Binutils的本机构建，所以设置这些变量可确保构建系统使用交叉编译器和关联的工具，而不是主机系统上的工具。

--with-lib-path=/tools/lib
这告诉配置脚本在Binutils的编译过程中指定库搜索路径，从而将/tools/lib 其传递给链接器。这样可以防止链接程序搜索主机上的库目录。

--with-sysroot
sysroot功能使链接器可以找到链接器命令行中明确包含的其他共享对象所需的共享对象。否则，某些软件包可能无法在某些主机上成功构建。

编译软件包：

```
make
```

安装软件包：

```
make install
```

现在，在下一章中为“ 重新调整 ”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
这告诉make程序删除ld 子目录中的所有已编译文件。

-C ld LIB_PATH=/usr/lib:/lib
此选项将重建ld子目录中的所有内容。LIB_PATH在命令行上指定 Makefile变量使我们可以覆盖临时工具的默认值，并将其指向正确的最终路径。此变量的值指定链接器的默认库搜索路径。下一章将使用此准备。

有关此软件包的详细信息，请参见第6.16.2节“ Binutils的内容”。

5.10。GCC-9.2.0-通过2

GCC软件包包含GNU编译器集合，其中包括C和C ++编译器。

时间：	预计编制 15 SBU

所需磁盘空间： 3.7 GB

5.10.1。安装GCC

我们的第一个GCC版本已经安装了几个内部系统头文件。通常，其中一个limits.h会依次包含相应的系统limits.h 标头，在这种情况下为/tools/include/limits.h。但是，在第一次构建gcc时/tools/include/limits.h不存在，因此GCC安装的内部标头是一个独立的部分文件，并且不包含系统标头的扩展功能。这足以构建临时的libc，但是现在构建的GCC需要完整的内部标头。使用与GCC构建系统在正常情况下相同的命令创建内部标头的完整版本：

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

再次更改GCC的默认动态链接器的位置，以使用中安装的链接器/tools。

```
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
```

如果基于x86_64构建，请将64位库的默认目录名称更改为“ lib ”：

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

与第一个GCC版本一样，它需要GMP，MPFR和MPC软件包。解压缩tarball，并将其移至所需的目录名称中：

```
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
```

再次创建一个单独的构建目录：

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

在开始构建GCC之前，请记住取消设置任何覆盖默认优化标志的环境变量。

现在准备GCC进行编译：

```
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
```

新的配置选项的含义：

--enable-languages=c,c++
此选项可确保同时构建C和C ++编译器。

--disable-libstdcxx-pch
不要为编译预编译的标头（PCH） `libstdc++`。它占用了大量空间，我们没有用。

--disable-bootstrap
对于GCC的本机版本，默认为“引导”版本。这不仅可以编译GCC，还可以编译几次。它使用在第一轮中编译的程序进行第二次自身编译，然后再次进行第三次编译。比较第二和第三次迭代，以确保它可以完美地自我复制。这也意味着它已正确编译。但是，LFS构建方法应提供可靠的编译器，而无需每次都进行引导。

编译软件包：

```
make
```

安装软件包：

```
make install
```

最后，创建一个符号链接。许多程序和脚本运行`cc` 而不是`gcc`，后者用于保持程序通用，因此可在并非总是安装GNU C编译器的各种UNIX系统上使用。运行`cc`使系统管理员可以自由决定要安装哪个C编译器：

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

警告

在这一点上，必须停止并确保新工具链的基本功能（编译和链接）按预期工作。要执行健全性检查，请运行以下命令：

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

如果一切正常，则应该没有错误，并且最后一条命令的输出将采用以下格式：

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

请注意，对于32位计算机，动态链接器为/tools/lib/ld-linux.so.2。

如果输出未如上所示或根本没有输出，则说明有问题。调查并追溯步骤以找出问题所在并纠正。必须继续解决此问题。首先，使用gcc而不是 cc再次执行完整性检查。如果/tools/bin/cc 可行，则符号链接丢失。按照上面的方法安装符号链接。接下来，确保PATH正确。可以通过运行echo $ PATH并验证其/tools/bin是否位于列表的开头来进行检查。如果PATH错误的消息，可能意味着您没有以用户身份登录， *lfs*或者在第4.4节"设置环境"中出了点问题。

一旦一切顺利，请清理测试文件：

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

有关此软件包的详细信息，请参见第6.21.2节" GCC的内容"。

5.11。Tcl-8.6.9

Tcl软件包包含工具命令语言。

	预计编制
时间：	0.9 SBU
所需磁盘空间：	71 MB

5.11.1。安装Tcl

安装此软件包以及接下来的两个软件包（Expect和DejaGNU）以支持运行GCC和Binutils以及其他软件包的测试套件。为测试目的安装三个软件包可能看起来过多，但要确保最重要的工具可以正常工作，即使不是必须的，也可以让您放心。即使本章未运行测试套件（它们不是强制性的），也需要这些软件包才能运行第6章中的测试套件。

请注意，此处使用的Tcl软件包是运行LFS测试所需的最低版本。有关完整软件包，请参见 **BLFS Tcl过程**。

准备编译Tcl：

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

构建包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Tcl测试套件，请发出以下命令：

```
TZ=UTC make test
```

在某些主机条件下，Tcl测试套件可能会发生故障，但尚不完全了解。因此，此处的测试套件故障不足为奇，也不被认为是至关重要的。该*TZ=UTC*参数将时区设置为协调世界时（UTC），但仅在测试套件运行期间。这样可以确保时钟测试正确执行。第7章TZ提供了有关环境变量的 详细信息。

安装软件包：

```
make install
```

使安装的库可写，以便以后可以删除调试符号：

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

安装Tcl的标头。下一个软件包Expect要求他们进行构建。

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

现在进行必要的符号链接：

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

5.11.2。Tcl的内容

已安装程序： tclsh（链接到tclsh8.6）和tclsh8.6

已安装的库： libtcl8.6.so，libtclstub8.6.a

简短说明

tclsh8.6	Tcl命令外壳
tclsh	到tclsh8.6的链接
libtcl8.6.so	Tcl库
libtclstub8.6.a	Tcl存根库

5.12。预期-5.45.4

Expect程序包包含一个程序，用于与其他交互式程序进行脚本化对话。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	4.0 MB

5.12.1。安装期望

首先，强制使用Expect的configure脚本，/bin/stty而不是/usr/local/bin/stty在主机系统上找到的脚本。这将确保我们的测试套件工具在工具链的最终构建中保持健全：

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

现在准备Expect进行编译：

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

配置选项的含义：

--with-tcl=/tools/lib
这样可以确保configure脚本在临时工具位置找到Tcl安装，而不是在主机系统上找到现有的Tcl安装。

--with-tclinclude=/tools/include
这明确地告诉Expect在哪里可以找到Tcl的内部标头。使用此选项可避免配置失败的情况，因为配置无法自动发现Tcl标头的位置。

构建包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Expect测试套件，请发出以下命令：

```
make test
```

请注意，Expect测试套件在某些主机条件下会出现故障，这些条件不在我们的控制范围之内。因此，此处的测试套件故障不足为奇，也不被认为是至关重要的。

安装软件包：

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

make参数的含义：

SCRIPTS=""
这样可以防止安装不需要的补充Expect脚本。

5.12.2。期望的内容

安装程序： 期望
已安装的库： libexpect-5.45.so

简短说明

期望	根据脚本与其他交互式程序进行通信
libexpect-5.45.so	包含允许将Expect用作Tcl扩展或直接从C或C ++使用的功能（不带Tcl）

5.13。DejaGNU-1.6.2

DejaGNU软件包包含用于测试其他程序的框架。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	3.2 MB

5.13.1。安装DejaGNU

准备DejaGNU进行编译：

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

构建并安装软件包：

```
make install
```

要测试结果，请发出：

```
make check
```

5.13.2。DejaGNU的内容

已安装程序： runtest

简短说明

运行测试 定位正确的一个包装脚本， 预计 壳，然后运行的DejaGnu

5.14。M4-1.4.18

M4软件包包含一个宏处理器。

预计编制
时间： 0.2 SBU
所需磁盘空间： 20 MB

5.14.1。安装M4

首先，进行glibc-2.28所需的一些修复：

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

准备M4进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行M4测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.14.2节“ M4的内容”。

5.15。Ncurses-6.1

Ncurses软件包包含用于终端独立处理字符屏幕的库。

	预计编制
时间：	0.6 SBU
所需磁盘空间：	41 MB

5.15.1。Ncurses的安装

首先，请确保在配置过程中首先找到gawk：

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

准备Ncurses进行编译：

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

配置选项的含义：

--without-ada
这样可以确保Ncurses不构建对主机上可能存在的Ada编译器的支持，但是一旦我们进入chroot环境就将不可用。

--enable-overwrite

这告诉Ncurses将其头文件安装到 /tools/include而不是中/tools/include/ncurses，以确保其他软件包可以成功找到Ncurses头。

--enable-widec

此切换导致libncursesw.so.6.1构建宽字符库（例如）而不是普通库（例如 libncurses.so.6.1）。这些宽字符库可用于多字节和传统的8位语言环境，而普通库仅可在8位语言环境中正常工作。宽字符库和普通库是源兼容的，但不是二进制兼容的。

编译软件包：

```
make
```

该软件包具有一个测试套件，但是只能在安装了该软件包之后才能运行。测试位于 test/目录中。有关README更多详细信息，请参见该目录中的 文件。

安装软件包：

```
make install
ln -s libncursesw.so /tools/lib/libncurses.so
```

有关此软件包的详细信息，请参见第6.24.2节“ Ncurses的内容”。

5.16。Bash-5.0

Bash软件包包含Bourne-Again SHell。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	67 MB

5.16.1。Bash的安装

准备Bash进行编译：

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

配置选项的含义：

--without-bash-malloc

此选项关闭了Bash的内存分配（ malloc ）函数的使用，该函数已知会导致分段错误。通过关闭此选项，Bash将使用 mallocGlibc中更稳定的功能。

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Bash测试套件，请发出以下命令：

```
make tests
```

安装软件包：

```
make install
```

为使用sh作为shell 的程序建立链接：

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

有关此软件包的详细信息，请参见第6.34.2节“ Bash的内容”。

5.17。野牛3.4.1

Bison软件包包含一个解析器生成器。

	预计编制
时间：	0.3 SBU
所需磁盘空间：	39 MB

5.17.1。野牛的安装

准备Bison进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.31.2节“野牛的内容”。

5.18。Bzip2-1.0.8

Bzip2软件包包含用于压缩和解压缩文件的程序。与传统的gzip相比，使用bzip2压缩文本文件产生的压缩百分比要好得多。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	6.0 MB

5.18.1。安装Bzip2

Bzip2软件包不包含配置脚本。使用以下命令进行编译和测试：

```
make
```

安装软件包：

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

有关此软件包的详细信息，请参见第6.22.2节“ Bzip2的内容”。

5.19。Coreutils-8.31

Coreutils软件包包含用于显示和设置基本系统特征的实用程序。

	预计编制
时间：	0.8 SBU
所需磁盘空间：	157 MB

5.19.1。安装Coreutils

准备Coreutils进行编译：

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

配置选项的含义：

--enable-install-program=hostname
这样就可以构建和安装主机名二进制文件-默认情况下它是禁用的，但Perl测试套件必需。

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Coreutils测试套件，请发出以下命令：

```
make RUN_EXPENSIVE_TESTS=yes check
```

该*RUN_EXPENSIVE_TESTS=yes* 参数告诉测试套件运行一些其他测试，这些测试在某些平台上被认为相对昂贵（就CPU能力和内存使用而言），但是通常在Linux上不是问题。

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.54.2节“ Coreutils的内容”。

5.20。Diffutils-3.7

Diffutils软件包包含显示文件或目录之间差异的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	26 MB

5.20.1。安装Diffutils

准备Diffutils进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Diffutils测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.56.2节“ Diffutils的内容”。

5.21。文件-5.37

文件包包含用于确定给定文件或多个文件类型的实用程序。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	19 MB

5.21.1。文件安装

准备要编译的文件：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行文件测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.12.2节“文件的内容”。

5.22。Findutils-4.6.0

Findutils程序包包含用于查找文件的程序。提供这些程序是为了在目录树中进行递归搜索并创建，维护和搜索数据库（通常比递归查找要快，但是如果最近未更新数据库则不可靠）。

	预计编制
时间：	0.3 SBU
所需磁盘空间：	36 MB

5.22.1。安装Findutils

首先，进行glibc-2.28所需的一些修复：

```
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
```

准备Findutils进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Findutils测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.58.2节“ Findutils的内容”。

5.23。高克5.0.1

Gawk软件包包含用于处理文本文件的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	46 MB

5.23.1。安装Gawk

准备Gawk进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Gawk测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.57.2节“ Gawk的内容”。

5.24。Gettext-0.20.1

Gettext程序包包含用于国际化和本地化的实用程序。这些允许程序使用NLS（本机语言支持）进行编译，从而使它们能够以用户的母语输出消息。

	预计编制
时间：	1.8 SBU
所需磁盘空间：	300 MB

5.24.1。Gettext的安装

对于我们的临时工具集，我们只需要从Gettext安装三个程序。

准备要编译的Gettext：

```
./configure --disable-shared
```

configure选项的含义：

--disable-shared
我们现在不需要安装任何共享的Gettext库，因此无需构建它们。

编译软件包：

```
make
```

由于环境有限，建议不要在此阶段运行测试套件。

安装msgfmt，msgmerge和xgettext程序：

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

有关此软件包的详细信息，请参见第6.47.2节“ Gettext的内容”。

5.25。Grep-3.3

Grep软件包包含用于搜索文件的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	24 MB

5.25.1。安装Grep

准备Grep进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Grep测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.33.2节“ Grep的内容”。

5.26。Gzip 1.10

Gzip软件包包含用于压缩和解压缩文件的程序。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	10 MB

5.26.1。安装Gzip

准备Gzip进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Gzip测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.62.2节“ Gzip的内容”。

5.27。制作4.2.1

Make软件包包含一个用于编译软件包的程序。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	13 MB

5.27.1。安装品牌

首先，解决由glibc-2.27和更高版本引起的错误：

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

准备进行编译：

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

configure选项的含义：

--without-guile
这样可以确保Make-4.2.1不会链接到主机系统上可能存在的Guile库，但在下一章的chroot环境中将不可用。

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Make测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.66.2节“制造内容”。

5.28。补丁2.7.6

补丁程序包包含一个程序，该程序用于通过应用 通常由diff程序创建的“ 补丁程序 ”文件来修改或创建文件。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	13 MB

5.28.1。安装补丁

准备补丁进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行补丁测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.67.2节“修补程序的内容”。

5.29。Perl-5.30.0

Perl软件包包含实用的提取和报告语言。

	预计编制
时间：	1.6 SBU
所需磁盘空间：	275 MB

5.29.1。安装Perl

准备要编译的Perl：

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

配置选项的含义：

-des
这是三个选项的组合：-d对所有项目使用默认值；-e确保完成所有任务；-s使非必要输出静音。

-Uloclibpth amd -Ulocincpth
这些条目未定义变量，这些变量使配置搜索主机系统上可能存在的本地安装的组件。

构建包：

```
make
```

尽管Perl带有测试套件，但是最好等到下一章中安装它。

此时仅需要安装一些实用程序和库：

```
cp -v perl cpan/podlators/scripts/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.30.0
cp -Rv lib/* /tools/lib/perl5/5.30.0
```

有关此软件包的详细信息，请参见第6.40.2节“ Perl的内容”。

5.30。Python-3.7.4

Python 3软件包包含Python开发环境。它对于面向对象的编程，编写脚本，对大型程序进行原型设计或开发整个应用程序很有用。

	预计编制
时间：	1.4 SBU
所需磁盘空间：	381 MB

5.30.1。安装Python

> **注意**
>
> 有两个名称以“ python ”开头的软件包文件 。要提取的是Python-3.7.4.tar.xz （请注意大写的第一个字母）。

该软件包首先构建Python解释器，然后构建一些标准Python模块。构建模块的主要脚本是用Python编写的，并使用到主机/usr/include和/usr/lib目录的硬编码路径 。为了防止使用它们，请发出：

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

准备要编译的Python：

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

configure选项的含义：

--without-ensurepip
此开关禁用Python软件包安装程序，此阶段不需要。

编译软件包：

```
make
```

编译完成。该测试套件需要TK和X Windows，目前无法运行。

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.51.2节“ Python 3的内容”。

5.31。Sed-4.7

Sed程序包包含一个流编辑器。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	20 MB

5.31.1。安装Sed

准备Sed进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Sed测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.28.2节“ Sed的内容”。

5.32。焦油1.32

Tar软件包包含一个归档程序。

	预计编制
时间：	0.3 SBU
所需磁盘空间：	38 MB

5.32.1。安装焦油

准备Tar以进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Tar测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.69.2节“ Tar的内容”。

5.33。Texinfo-6.6

Texinfo软件包包含用于读取，写入和转换信息页面的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	103 MB

5.33.1。安装Texinfo

准备Texinfo进行编译：

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

> **注意**
>
> 作为配置过程的一部分，将进行测试以指示TestXS_la-TestXS.lo错误。这与LFS无关，应该忽略。

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Texinfo测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.70.2节“ Texinfo的内容”。

5.34。Xz-5.2.4

Xz软件包包含用于压缩和解压缩文件的程序。它为lzma和更新的xz压缩格式提供功能。与传统的gzip或 bzip2命令相比，使用xz压缩文本文件产生的压缩百分比更高 。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	18 MB

5.34.1。安装Xz

准备Xz进行编译：

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

编译软件包：

```
make
```

编译完成。如前所述，对于本章中的临时工具，运行测试套件不是必需的。无论如何要运行Xz测试套件，请发出以下命令：

```
make check
```

安装软件包：

```
make install
```

有关此软件包的详细信息，请参见第6.45.2节“ Xz的内容”。

5.35。剥离

本节中的步骤是可选的，但是如果LFS分区很小，则了解可以删除不必要的项目将很有帮助。到目前为止构建的可执行文件和库包含大约70 MB的不需要的调试符号。使用以下方法删除这些符号：

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

这些命令将跳过许多文件，报告它无法识别其文件格式。其中大多数是脚本而不是二进制文件。还可以使用system strip命令在/ tools中包含strip二进制文件。

注意*不要*`--strip-unneeded`在库上使用 。静态软件包将被销毁，并且需要重新构建工具链软件包。

要保存更多，请删除文档：

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

删除不需要的文件：

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

此时，您应该至少有3 GB的可用空间 $LFS可用于在下一阶段中构建和安装Glibc和Gcc。如果可以构建和安装Glibc，则也可以构建和安装其余的Glibc。

5.36。改变所有权

> **注意**
>
> 本书其余部分中的命令必须在以用户身份登录*root* 且不再以用户身份登录时执行*lfs*。另外，请仔细检查$LFS在*root*的环境中设置的内容。

当前，$LFS/tools 目录归用户所有，该用户*lfs*仅存在于主机系统上。如果$LFS/tools 目录保持不变，则文件由用户ID拥有，而没有相应的帐户。这很危险，因为以后创建的用户帐户可能会获得相同的用户ID，并拥有$LFS/tools 目录和其中的所有文件，从而使这些文件容易受到恶意操纵。

为避免此问题，您可以*lfs*稍后在创建/etc/passwd文件时将用户添加到新的LFS系统中，请注意为其分配与主机系统上相同的用户和组ID。更好的是，通过运行以下命令将$LFS/tools目录的所有权更改 为user *root*：

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

尽管$LFS/tools可以在LFS系统完成后删除该目录，但是可以保留该目录以构建*相同书籍版本的*其他LFS系统。如何最好地备份$LFS/tools是个人喜好的问题。

> **警告**
>
> 如果您打算保留临时工具以用于构建未来的LFS系统，***现在***是时候对其进行备份了。第6章中的后续命令将更改当前使用的工具，从而使它们对将来的构建毫无用处。

第三部分 建立LFS系统

第6章。安装基本系统软件

6.1。介绍

在本章中，我们进入建筑工地并开始认真构建LFS系统。也就是说，我们将chroot放入临时的迷你Linux系统中，进行一些最后的准备，然后开始安装软件包。

该软件的安装非常简单。尽管在许多情况下，安装说明可以变得更短，更通用，但我们选择为每个软件包提供完整的说明，以最大程度地减少出错的可能性。学习使Linux系统正常工作的关键是知道每个软件包的用途以及为什么您（或系统）需要它。

我们不建议使用优化。它们可以使程序的运行速度稍快一些，但在运行程序时也可能导致编译困难和问题。如果软件包在使用优化时拒绝编译，请尝试在不进行优化的情况下进行编译，看看是否可以解决问题。即使程序包在使用优化时确实可以编译，由于代码和构建工具之间的复杂交互作用，也可能存在编译错误的风险。还要注意的是，`-march`和`-mtune` 使用书中未指定值的选项尚未经过测试。这可能会导致工具链包（Binutils，GCC和Glibc）出现问题。使用编译器优化所获得的少量潜在收益通常会被风险所抵消。鼓励LFS的初次构建者在没有自定义优化的情况下进行构建。后续系统仍将运行非常快，并且同时稳定。

必须严格遵守本章中软件包的安装顺序，以确保没有程序意外地获得指向`/tools`硬连线到其中的路径。由于相同的原因，请勿并行编译单独的软件包。并行编译可以节省时间（尤其是在双CPU机器上），但是它可能导致程序包含到的硬连线路径`/tools`，这将导致该程序在删除该目录时停止工作。

在安装说明之前，每个安装页面都会提供有关软件包的信息，包括对其内容的简要说明，大约需要花费多长时间来构建以及在此构建过程中需要多少磁盘空间。按照安装说明进行操作，并列出了软件包将安装的程序和库的列表（以及对它们的简要说明）。

> **注意**
>
> SBU值和所需的磁盘空间包括第6章中所有适用软件包的测试套件数据。

6.1.1。关于图书馆

通常，LFS编辑器不鼓励构建和安装静态库。在现代Linux系统中，大多数静态库的原始目的已经过时。另外，将静态库链接到程序中可能是有害的。如果需要更新库以消除安全性问题，则所有使用静态库的程序都需要重新链接到新库。由于静态库的使用并不总是很明显，因此可能甚至不知道相关程序（以及进行链接所需的过程）。

在第6章中的过程中，我们删除或禁用大多数静态库的安装。通常，这是通过传递一个`配置--disable-static` 选项来完成的。在其他情况下，则需要其他方法。在某些情况下，尤其是glibc和gcc，静态库的使用对于一般的程序包构建过程仍然至关重要。

有关库的更完整讨论，请参见 **库：静态还是共享？**在BLFS书中。

6.2。准备虚拟内核文件系统

内核导出的各种文件系统用于与内核本身进行通信。这些文件系统是虚拟的，因为它们没有使用磁盘空间。文件系统的内容驻留在内存中。

首先创建将文件系统挂载到的目录：

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

6.2.1。创建初始设备节点

当内核启动系统时，它需要存在一些设备节点，尤其是console and null设备。必须在硬盘上创建设备节点，以便在启动udevd之前 以及在使用Linux启动时可以使用它们 *init=/bin/bash*。通过运行以下命令来创建设备：

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

6.2.2。安装和填充/ dev

建议/dev使用设备填充目录的方法是在目录上安装虚拟文件系统（例如 *tmpfs* ）/dev，并允许在检测到或访问设备时在该虚拟文件系统上动态创建设备。设备创建通常由Udev在引导过程中完成。由于此新系统尚没有Udev并且尚未启动，因此必须/dev手动安装和填充。这是通过绑定安装主机系统的/dev 目录。绑定安装是一种特殊的安装类型，它允许您创建目录或安装点到其他位置的镜像。使用以下命令可实现此目的：

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

6.2.3。挂载虚拟内核文件系统

现在挂载剩余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
```

devpts的安装选项的含义：

gid=5

这样可以确保所有由devpts创建的设备节点都归组ID 5拥有。这是我们稍后将用于该 *tty* 组的ID 。我们使用组ID代替名称，因为主机系统可能为其 *tty* 组使用其他ID 。

mode=0620

这样可以确保所有由devpts创建的设备节点都具有模式0620（用户可读可写，组可写）。与上面的选项一起，这确保了devpts将创建满足grantpt（ ）要求的设备节点，这意味着不需要Glibc pt_chown 帮助程序二进制文件（默认情况下未安装）。

在某些主机系统中，/dev/shm 是的符号链接/run/shm。/ run tmpfs安装在上面，因此在这种情况下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。包装管理

软件包管理是LFS手册中经常需要添加的内容。包管理器允许跟踪文件的安装，从而轻松删除和升级包。除二进制文件和库文件外，程序包管理器还将处理配置文件的安装。在您开始怀疑之前，请不要，本节将不讨论也不推荐任何特定的软件包管理器。它提供了一些更流行的技术及其工作原理的综述。对于您来说，理想的软件包管理器可能是这些技术中的一种，或者可以是这些技术中的两种或多种的组合。本节简要提及升级软件包时可能出现的问题。

在LFS或BLFS中未提及包管理器的一些原因包括：

- 处理软件包管理使重点不再放在这些书的目标上，而是讲授如何构建Linux系统。
- 软件包管理有多种解决方案，每种都有其优点和缺点。要让所有观众都能满意就很难。

关于包管理的主题有一些提示。访问**提示项目**，看看是否其中之一适合您的需求。

6.3.1。升级问题

软件包管理器使发布新版本时轻松升级。通常，LFS和BLFS手册中的说明可用于升级到较新版本。在升级软件包时，尤其是在运行的系统上，应注意以下几点。

- 如果需要将Glibc升级到较新的版本（例如，从glibc-2.19升级到glibc-2.20），则重建LFS更安全。尽管您*可以*按依赖关系顺序重建所有软件包，但我们不建议这样做。
- 如果包含共享库的软件包已更新，并且库的名称发生了更改，则需要重新编译所有动态链接到该库的软件包，以针对较新的库进行链接。（请注意，程序包版本和库名之间没有关联。）例如，考虑一个安装了name共享库的foo-1.2.3程序包`libfoo.so.1`。假设您将该软件包升级到了较新的版本foo-1.2.4，该版本将安装名为name的共享库`libfoo.so.2`。在这种情况下，所有动态链接到的软件包`libfoo.so.1`需要重新编译以链接到`libfoo.so.2`。请注意，在重新编译从属软件包之前，您不应删除以前的库。

6.3.2。包管理技术

以下是一些常见的软件包管理技术。在确定包管理器之前，请对各种技术进行一些研究，尤其是特定方案的缺点。

6.3.2.1。一切都在我的脑海！

是的，这是一种软件包管理技术。有些人并不需要软件包管理器，因为他们非常了解软件包，并且知道每个软件包安装了哪些文件。一些用户也不需要任何程序包管理，因为他们计划在更改程序包时重建整个系统。

6.3.2.2。在单独目录中安装

这是一种简化的软件包管理，不需要任何额外的软件包即可管理安装。每个软件包都安装在单独的目录中。例如，在其中安装了软件包foo-1.1，`/usr/pkg/foo-1.1`并从`/usr/pkg/foo`到 进行了符号链接`/usr/pkg/foo-1.1`。安装新版本的foo-1.2时，会安装它，`/usr/pkg/foo-1.2`并且以前的符号链接将替换为新版本的符号链接。

环境变量，例如`PATH`，`LD_LIBRARY_PATH`，`MANPATH`，`INFOPATH`并`CPPFLAGS`加以扩展需要包括`/usr/pkg/foo`。对于多个软件包，此方案变得难以管理。

6.3.2.3。Symlink样式包管理

这是以前的程序包管理技术的变体。每个软件包的安装方式与以前的方案类似。但是，不是进行符号链接，而是将每个文件符号链接到/usr层次结构中。这样就无需扩展环境变量。尽管用户可以创建符号链接来自动创建符号链接，但是已经使用这种方法编写了许多程序包管理器。一些受欢迎的工具包括Stow，Epkg，Graft和Depot。

需要伪造安装，以便该软件包认为它已安装在其中，/usr尽管实际上它已安装在/usr/pkg层次结构中。以这种方式安装通常不是一件容易的事。例如，假设您正在安装软件包libfoo-1.1。以下说明可能无法正确安装该软件包：

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

安装将可以进行，但是从属软件包可能不会按您期望的那样链接到libfoo。如果您编译了一个链接到libfoo的程序包，您可能会注意到它是链接到的，/usr/pkg/libfoo/1.1/lib/libfoo.so.1 而不是/usr/lib/libfoo.so.1您期望的那样。正确的方法是使用该DESTDIR策略来伪造软件包的安装。此方法的工作方式如下：

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

大多数软件包都支持这种方法，但是有些不支持。对于不兼容的软件包，您可能需要手动安装该软件包，或者可能会发现将一些有问题的软件包安装到会更容易/opt。

6.3.2.4。基于时间戳

在此技术中，在安装软件包之前先给文件加上时间戳。安装后，简单地使用带有适当选项的find命令可以生成创建时间戳文件之后安装的所有文件的日志。用这种方法编写的程序包管理器是install-log。

尽管该方案具有简单的优点，但是它具有两个缺点。如果在安装过程中使用当前时间以外的任何时间戳来安装文件，则程序包管理器将不会跟踪这些文件。此外，仅当一次安装一个软件包时，才可以使用此方案。如果在两个不同的控制台上安装了两个软件包，则日志不可靠。

6.3.2.5。跟踪安装脚本

通过这种方法，将记录安装脚本执行的命令。可以使用两种技术：

该LD_PRELOAD环境变量可以设置为指向库安装之前预加载。在安装过程中，该库通过将自身附加到各种可执行文件（例如cp，install，mv）来跟踪正在安装的软件包并跟踪修改文件系统的系统调用。为了使这种方法起作用，所有可执行文件都需要动态链接而无需suid或sgid位。预加载库可能会在安装过程中产生一些不良影响。因此，建议执行一些测试以确保程序包管理器不会破坏任何内容并记录所有适当的文件。

第二种技术是使用strace，它记录在安装脚本执行期间进行的所有系统调用。

6.3.2.6。创建软件包档案

在此方案中，如Symlink样式的软件包管理中所述，将软件包安装伪造到单独的树中。安装后，将使用已安装的文件创建软件包归档文件。然后，此归档文件用于在本地计算机上安装软件包，甚至可以用于在其他计算机上安装软件包。

商业发行版中的大多数程序包管理器都使用这种方法。遵循这种方法的软件包管理器的示例包括RPM（**Linux标准基础规范**偶然需要），pkg-utils，Debian的apt和Gentoo的Portage系统。 **http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt上有**一个提示，描述如何为LFS系统采用这种软件

包管理风格。

包含依赖项信息的软件包文件的创建很复杂，并且超出了LFS的范围。

Slackware使用基于tar的系统来打包软件包。该系统故意不像更复杂的软件包管理器那样处理软件包依赖性。有关Slackware软件包管理的详细信息，请参见 **http://www.slackbook.org/html/package-management.html**。

6.3.2.7。基于用户的管理

该方案是LFS特有的，由Matthias Benkmann设计，可以从**Hints Project中获得**。在此方案中，每个软件包都以单独的用户身份安装到标准位置。通过检查用户ID可以轻松识别属于软件包的文件。此方法的功能和缺点太复杂，无法在本节中进行描述。有关详细信息，请参见 **http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt中**的提示。

6.3.3。在多个系统上部署LFS

LFS系统的优点之一是没有文件依赖于磁盘系统上文件的位置。将LFS构建克隆到与基础系统具有相同体系结构的另一台计算机，就像在包含根目录的LFS分区上使用tar一样简单（对于基本LFS构建，约有250MB未压缩），然后通过网络传输或CD-复制该文件ROM到新系统并进行扩展。从那时起，将必须更改一些配置文件。可能需要更新配置文件包括：/etc/hosts， /etc/fstab，/etc/passwd，/etc/group，/etc/shadow， /etc/ld.so.conf，/etc/sysconfig/rc.site，/etc/sysconfig/network，和/etc/sysconfig/ifconfig.eth0。

根据系统硬件和原始内核配置的差异，可能需要为新系统构建自定义内核。

> **注意**
>
> 在相似但不相同的体系结构之间进行复制时，有一些关于问题的报告。例如，用于Intel系统的指令集与AMD处理器不相同，某些处理器的更高版本可能具有在早期版本中不可用的指令。

最后，必须通过第8.4节"使用GRUB设置启动过程"使新系统可启动。

6.4。进入Chroot环境

现在是时候进入chroot环境开始构建和安装最终的LFS系统了。以user *root*身份，运行以下命令以输入当前仅由临时工具填充的领域：

```
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
```

*-i*给env命令 的选项将清除chroot环境的所有变量。在那之后，只有HOME，TERM，PS1，和 PATH变量重新设置。该 *TERM=$TERM*构造会将TERMchroot内部的变量设置为与chroot外部的变量相同的值。vim之类的程序需要此变量，而less才能正常运行。如果需要其他变量，例如CFLAGS或CXXFLAGS，这是再次设置它们的好地方。

从现在开始，不再需要使用该LFS变量，因为所有工作都将限于LFS文件系统。这是因为告诉Bash Shell $LFS现在是根（/）目录。

请注意，/tools/bin它位于中PATH。这意味着一旦安装了最终版本，便不再使用临时工具。当shell无法“ 记住 ”已执行的二进制文件的位置时，就会发生这种情况一因此，通过将*+h*选项传递给bash来关闭哈希。

请注意，bash提示符将说 I have no name!这是正常现象，因为/etc/passwd 尚未创建文件。

> **注意**
>
> 重要的是，本章其余部分和以下各章中的所有命令都必须在chroot环境中运行。如果出于任何原因（例如，重新引导）离开此环境，请确保按照第6.2.2节“安装和填充/ dev”和第6.2.3节“安装虚拟内核文件系统”中的说明安装虚拟内核文件系统。并在继续安装之前再次输入chroot。

6.5。创建目录

现在该在LFS文件系统中创建一些结构了。通过发出以下命令来创建标准目录树：

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib/firmware,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{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}
mkdir -v  /usr/lib/pkgconfig

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}
```

默认情况下，目录是使用权限模式755创建的，但这并不是所有目录都希望的。在上面的命令中，进行了两项更改-一项更改为user的主目录*root*，另一项更改为临时文件的目录。

第一次模式更改可确保不仅任何人都可以输入/root目录，这与普通用户对其主目录的处理方式相同。第二种模式更改可确保任何用户都可以写入/tmp和/var/tmp目录，但不能从其中删除另一个用户的文件。后者被所谓的" 粘性位 "禁止使用，即1777位掩码中的最高位（1）。

6.5.1。FHS合规说明

目录树基于文件系统层次结构标准（FHS）（可从**https://refspecs.linuxfoundation.org/fhs.shtml获得**）。FHS还指定了某些目录（例如/usr/local/games和）的可选存在/usr/share/games。我们仅创建所需的目录。但是，请随意创建这些目录。

6.6。创建基本文件和符号链接

一些程序使用到尚不存在的程序的硬连线路径。为了满足这些程序的要求，请创建许多符号链接，这些符号链接在安装软件后的本章中将被实际文件替换：

```
ln -sv /tools/bin/{bash,cat,chmod,dd,echo,ln,mkdir,pwd,rm,stty,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++.{a,so{,.6}}             /usr/lib

ln -sv bash /bin/sh
```

每个链接的目的：

/bin/bash
许多bash 脚本指定/bin/bash。

/bin/cat
该路径名被硬编码到Glibc的configure脚本中。

/bin/dd
的路径dd将被硬编码到/usr/bin/libtool实用程序中。

/bin/echo
这是为了满足Glibc测试套件中的其中一项测试/bin/echo。

/usr/bin/env
该路径名被硬编码到某些软件包的构建过程中。

/usr/bin/install
的路径install将被硬编码到/usr/lib/bash/Makefile.inc文件中。

/bin/ln
的路径ln将被硬编码到/usr/lib/perl5/5.30.0/<target-triplet>/Config_heavy.pl 文件中。

/bin/pwd
一些配置脚本（尤其是Glibc脚本）具有此路径名的硬编码。

/bin/rm
的路径rm将被硬编码到/usr/lib/perl5/5.30.0/<target-triplet>/Config_heavy.pl 文件中。

/bin/stty
该路径名被硬编码到Expect中，因此Binutils和GCC测试套件必须通过。

/usr/bin/perl
许多Perl脚本将此路径硬编码为perl程序。

/usr/lib/libgcc_s.so{,.1}
Glibc需要使用它来使pthreads库起作用。

/usr/lib/libstdc++{,.6}
Glibc测试套件中的一些测试以及GMP中的C ++支持都需要这样做。

/bin/sh
许多shell脚本是硬编码的/bin/sh。

历史上，Linux在file中维护已挂载文件系统的列表/etc/mtab。现代内核在内部维护此列表，并通过/proc 文件系统将其公开给用户。为了满足期望使用的实用程序 /etc/mtab，请创建以下符号链接：

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

为了使用户*root*能够登录并识别名称" root "，/etc/passwdand /etc/group文件中必须有相关条目 。

/etc/passwd通过运行以下命令来 创建文件：

```
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
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

root （此处使用的" x "只是一个占位符） 的实际密码将在以后设置。

/etc/group通过运行以下命令来 创建文件：

```
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:
input:x:24:
mail:x:34:
kvm:x:61:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

创建的组不是任何标准的一部分，它们是由本章中Udev配置的要求以及部分由许多现有Linux发行版采用的通用约定决定的组。另外，某些测试套件依赖于特定的用户或组。Linux Standard Base（LSB，可从**http://www.linuxbase.org上获得**）仅建议除了*root*组ID（GID）为0的组之外， *bin*的GID为1。所有其他的组名和GID可以由系统管理员自由选择，因为编写良好的程序并不依赖于GID号，而是使用组名。

要删除" 我没有名字！"提示，启动新的外壳。由于在第5章中安装了完整的Glibc并创建 了/etc/passwd和 /etc/group文件，因此现在可以使用用户名和组名解析：

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

注意*+h* 指令的使用。这告诉bash不要使用其内部路径哈希。如果没有该指令，bash将记住它已执行的二进制文件的路径。为了确保在安装新编译的二进制文件后立即使用它们，在*+h*本章中将使用伪指令。

在登录， 的agetty，和 初始化程序（和其他人）使用一些日志文件来记录信息，比如谁登录到系统和时间。但是，如果这些程序尚不存在，它们将不会写入日志文件。初始化日志文件并为其赋予适当的权限：

```
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
```

该/var/log/wtmp文件记录所有登录和注销。该/var/log/lastlog文件记录每个用户上次登录的时间。该/var/log/faillog文件记录失败的登录尝试。该/var/log/btmp文件记录错误的登录尝试。

注意

该/run/utmp文件记录了当前登录的用户。该文件是在启动脚本中动态创建的。

6.7。Linux-5.2.8 API标头

Linux API标头（在linux-5.2.8.tar.xz中）公开了内核的API，供Glibc使用。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	960 MB

6.7.1。安装Linux API标头

Linux内核需要公开一个应用程序编程接口（API），供系统的C库（LFS中的Glibc）使用。这是通过清理Linux内核源tarball中附带的各种C头文件来完成的。

确保之前的活动没有陈旧的文件和依赖项：

```
make mrproper
```

现在从源代码中提取用户可见的内核头文件。它们被放置在中间本地目录中并复制到所需的位置，因为提取过程会删除目标目录中的所有现有文件。从中间目录中删除的还有一些内核开发人员使用的隐藏文件，LFS不需要这些文件。

```
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标头的内容

安装的标头：	/ 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
安装目录：	/ usr / include / asm，/ usr / include / asm-generic，/ usr / include / drm，/ usr / include / linux，/ usr / include / misc，/ usr / include / mtd，/ usr /包括/ rdma，/ usr / include / scsi，/ usr / include / sound，/ usr / include / video和/ usr / include / xen

简短说明

/usr/include/asm/*.h	Linux API ASM标头
/usr/include/asm-generic/*.h	Linux API ASM通用标头
/usr/include/drm/*.h	Linux API DRM标头
/usr/include/linux/*.h	Linux API Linux标头
/usr/include/misc/*.h	Linux API杂项标头
/usr/include/mtd/*.h	Linux API MTD标头
/usr/include/rdma/*.h	Linux API RDMA标头
/usr/include/scsi/*.h	Linux API SCSI标头
/usr/include/sound/*.h	Linux API声音标题
/usr/include/video/*.h	Linux API视频标题
/usr/include/xen/*.h	Linux API Xen标头

6.8。手册页5.02

手册页包包含2200多个手册页。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	31 MB

6.8.1。手册页的安装

通过运行以下命令安装手册页：

```
make install
```

6.8.2。手册页的内容

安装的文件： 各种手册页

简短说明

man pages	描述C编程语言功能，重要的设备文件和重要的配置文件

6.9。Glibc-2.30

Glibc软件包包含主要的C库。该库提供了用于分配内存，搜索目录，打开和关闭文件，读取和写入文件，字符串处理，模式匹配，算术等等的基本例程。

	预计编制
时间：	21 SBU
所需磁盘空间：	3.3 GB

6.9.1。安装Glibc

> **注意**
>
> 即使编译器规范文件和链接器仍指向，Glibc构建系统都是独立的并且可以完美安装/tools。在安装Glibc之前，无法调整规范和链接器，因为Glibc自动配置测试会给出错误的结果，从而无法实现干净构建的目标。

一些Glibc程序使用非FHS编译 /var/db目录来存储其运行时数据。应用以下补丁以使此类程序将其运行时数据存储在符合FHS的位置：

```
patch -Np1 -i ../glibc-2.30-fhs-1.patch
```

修复linux-5.2内核引入的问题：

```
sed -i '/asm.socket.h/a# include <linux/sockios.h>' \
   sysdeps/unix/sysv/linux/bits/socket.h
```

创建符号链接以符合LSB。此外，对于x86_64，创建动态加载程序正常运行所需的兼容性符号链接：

```
case $(uname -m) in
    i?86)   ln -sfv ld-linux.so.2 /lib/ld-lsb.so.3
    ;;
    x86_64) 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
```

Glibc文档建议在专用的构建目录中构建Glibc：

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

准备Glibc进行编译：

```
CC="gcc -ffile-prefix-map=/tools=/usr" \
../configure --prefix=/usr                          \
             --disable-werror                       \
             --enable-kernel=3.2                    \
             --enable-stack-protector=strong        \
             --with-headers=/usr/include            \
             libc_cv_slibdir=/lib
```

选项和新的配置参数的含义：

CC="gcc -ffile-prefix-map=/tools=/usr"
使GCC在编译后记录/ tools中对文件的所有引用，就像文件位于/ usr中一样。这样可以避免在调试符号中引入无效路径。

--disable-werror
此选项禁用传递给GCC的-Werror选项。这对于运行测试套件是必需的。

--enable-stack-protector=strong
此选项通过添加额外的代码来检查缓冲区溢出（例如堆栈粉碎攻击）来提高系统安全性。

--with-headers=/usr/include
此选项告诉构建系统在哪里可以找到内核API标头。默认情况下，这些标头在中查找/tools/include。

libc_cv_slibdir=/lib
该变量为所有系统设置正确的库。我们不希望使用lib64。

编译软件包：

```
make
```

> **重要**
>
> 在本节中，Glibc的测试套件被认为是至关重要的。在任何情况下都不要跳过它。

通常，一些测试不会通过。通常可以忽略下面列出的测试失败。

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

> **注意**
>
> 在chroot环境中构建此阶段的阶段，需要上面的符号链接来运行测试。在下面的安装阶段它将被覆盖。

```
make check
```

您可能会看到一些测试失败。Glibc测试套件在某种程度上取决于主机系统。这是某些版本的LFS中最常见的问题列表：

- 已知 *misc / tst-ttyname*在LFS chroot环境中失败。
- 已知 *inet / tst-idna_name_classify*在LFS chroot环境中失败。
- *posix / tst-getaddrinfo4*和 *posix / tst-getaddrinfo5*在某些体系结构上可能会失败。
- 在*NSS / TST-NSS-文件的主机的多* 测试可能会失败，对于尚未确定的原因。
- 的*RT / TST-cputimer {1,2,3}*测试依赖于主机系统的内核。已知内核4.14.91–4.14.96、4.19.13–4.19.18和4.20.0–4.20.5会导致这些测试失败。
- 在CPU不是相对较新的Intel或AMD处理器的系统上运行时，数学测试有时会失败。

尽管这是一条无害的消息，但Glibc的安装阶段将抱怨缺少`/etc/ld.so.conf`。防止出现以下警告：

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

修复生成的Makefile，以跳过在LFS部分环境中失败的不必要的完整性检查：

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

安装软件包：

```
make install
```

安装`nscd`的配置文件和运行时目录 ：

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

接下来，安装可以使系统以其他语言响应的语言环境。不需要任何语言环境，但是如果缺少某些语言环境，则将来软件包的测试套件将跳过重要的测试用例。

可以使用`localedef`程序安装各个语言环境。例如，下面的第一个`localedef`命令将与`/usr/share/i18n/locales/cs_CZ` 字符集无关的语言环境定义与`/usr/share/i18n/charmaps/UTF-8.gz`charmap定义相结合，并将结果附加到`/usr/lib/locale/locale-archive`文件中。以下说明将安装最佳测试所必需的最小区域设置：

```
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
```

另外，为您自己的国家，语言和字符集安装语言环境。

或者，glibc-2.30/localedata/SUPPORTED使用以下耗时的命令一次安装文件中列出的所有语言环境（包括上面列出的每个语言环境以及更多其他语言）：

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

然后，在不太可能需要的情况下，使用localedef命令创建并安装glibc-2.30/localedata/SUPPORTED文件中未列出的语言环境。

> **注意**
>
> 现在，Glibc在解析国际化域名时使用libidn2。这是运行时依赖项。如果需要此功能，请在 **BLFS libidn2页面中**安装libidn2的说明。

6.9.2。配置Glibc

6.9.2.1。添加nsswitch.conf

/etc/nsswitch.conf由于Glibc默认设置在网络环境中无法正常运行，因此需要创建 该文件。

/etc/nsswitch.conf通过运行以下 命令创建一个新文件：

```
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。添加时区数据

使用以下方法安装和设置时区数据：

```
tar -xf ../../tzdata2019b.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
```

zic命令的含义：

zic -L /dev/null ...
这将创建posix时区，而没有任何leap秒。通常将它们都放在 zoneinfo和中 zoneinfo/posix。必须将POSIX时区设置为zoneinfo，否则各种测试套件都会报告错误。在空间有限且您不打算更新时区的嵌入式系统上，不使用posix 目录可以节省1.9MB ，但是某些应用程序或测试套件可能会导致某些故障。

zic -L leapseconds ...
这将创建正确的时区，包括leap秒。在空间狭小的嵌入式系统上，您不打算更新时区，也不在乎正确的时间，因此可以通过省略right 目录来节省1.9MB 。

zic ... -p ...
这将创建posixrules文件。我们之所以使用纽约，是因为POSIX要求夏令时规则与美国规则一致。

确定本地时区的一种方法是运行以下脚本：

```
tzselect
```

在回答了有关位置的几个问题之后，脚本将输出时区的名称（例如 *America / Edmonton* ）。还列出了一些其他可能的时区， /usr/share/zoneinfo例如 *加拿大/东部*或 *EST5EDT*，这些时区未由脚本标识，但可以使用。

然后/etc/localtime运行以下命令创建文件：

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

替换*<xxx>*为所选时区的名称（例如，加拿大/东部）。

6.9.2.3。配置动态加载程序

默认情况下，动态加载程序（ /lib/ld-linux.so.2 ）在程序运行时搜索 /lib并/usr/lib查找程序所需的动态库。但是，如果除/lib和以外的目录中都有库，则/usr/lib需要将这些库添加到 /etc/ld.so.conf文件中，以便动态加载程序找到它们。众所周知两个包含其他库的目录是 /usr/local/lib和 /opt/lib，因此请将这些目录添加到动态加载程序的搜索路径中。

/etc/ld.so.conf通过运行以下 命令创建一个新文件：

```
cat > /etc/ld.so.conf << "EOF"
# Begin /etc/ld.so.conf
/usr/local/lib
/opt/lib

EOF
```

如果需要，动态加载程序还可以搜索目录并包括在该目录中找到的文件的内容。通常，此包含目录中的文件是一行，指定所需的库路径。要添加此功能，请运行以下命令：

```
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。Glibc的内容

已安装程序： catchsegv，gencat，getconf，getent，iconv，iconvconfig，ldconfig，ldd，lddlibc4，语言环境，localedef，makedb，mtrace，nscd，pcprofiledump，pldd，sln，struruss，sprof，tzselect，xtrace，zdump和zic

已安装的库： ld-2.30.so，libBrokenLocale。{a，so}，libSegFault.so，libanl。{a，so}，libc。{a，so}，libc_nonshared.a，libcrypt。{a，so}，libdl 。{a，so}，libg.a，libm。{a，so}，libmcheck.a，libmemusage.so，libmvec。{a，so}，libnsl。{a，so}，libnss_compat.so，libnss_dns.so ，libnss_files.so，libnss_hesiod.so，libpcprofile.so，libpthread。{a，so}，libpthread_nonshared.a，libresolv。{a，so}，librt。{a，so}，libthread_db.so和libutil。{a ，所以}

安装目录： / 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 / sys，/ usr / lib / audit，/ usr / lib / gconv，/ usr / lib / locale， / usr / libexec / getconf，/ usr / share / i18n，/ usr / share / zoneinfo，/ var / cache / nscd和/ var / lib / nss_db

简短说明

catchsegv	当程序因分段错误而终止时，可用于创建堆栈跟踪
Gencat	生成消息目录
getconf	显示文件系统特定变量的系统配置值
Getent	从管理数据库获取条目
图标	执行字符集转换
iconvconfig	创建快速加载的 iconv 模块配置文件
ldconfig	配置动态链接器运行时绑定
dd	报告每个给定程序或共享库需要哪些共享库
lddlibc4	协助 LDD 处理目标文件
地区	打印有关当前语言环境的各种信息
本地环境	编译语言环境规范
制作数据库	根据文本输入创建一个简单的数据库
跟踪	读取和解释内存跟踪文件，并以人类可读的格式显示摘要

光盘	为最常见的名称服务请求提供缓存的守护程序
pcprofiledump	PC配置文件生成的转储信息
dd	列出正在运行的进程使用的动态共享对象
sln	静态链接的 ln 程序
tru	跟踪指定命令的共享库过程调用
骚扰	读取并显示共享对象分析数据
选择	向用户询问系统的位置并报告相应的时区描述
跟踪	通过打印当前执行的函数来跟踪程序的执行
转储	时区自卸车
奇克	时区编译器
ld-2.30.so	共享库可执行文件的帮助程序
libBrokenLocale	Glibc在内部用作总的黑客工具，以使损坏的程序（例如，某些Motif应用程序）运行。请参阅中的评论以glibc-2.30/locale/broken_cur_max.c 获取更多信息
libSegFault	catchsegv 使用的分段故障信号处理程序
libanl	异步名称查找库
libc	主要的C库
libcrypt	密码学图书馆
libdl	动态链接接口库
libg	虚拟库不包含任何函数。以前是 g ++ 的运行时库
libm	数学库
libmcheck	链接到时打开内存分配检查
libmemusage	通过使用 MEMUSAGE 帮助有关程序的内存使用情况收集信息
libnsl	网络服务库
libnss	名称服务交换库，包含用于解析主机名，用户名，组名，别名，服务，协议等的功能。
libpcprofile	可以预加载到PC配置文件中的可执行文件
libpthread	POSIX线程库
libresolv	包含用于创建，发送和解释数据包到Internet域名服务器的功能
librt	包含提供POSIX.1b Realtime Extension指定的大多数接口的函数
libthread_db	包含对构建多线程程序调试器有用的函数
libutil	包含用于许多不同Unix实用程序的“ 标准 ”功能的 代码

6.10。调整工具链

现在已经安装了最终的C库，是时候调整工具链了，以便它将所有新编译的程序与这些新库链接在一起。

首先，备份/tools链接器，并将其替换为我们在第5章中所做的调整后的链接器。我们还将在以下位置创建指向其对应对象的链接/tools/$(uname -m)-pc-linux-gnu/bin：

```
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规范文件，使其指向新的动态链接器。只需删除“ / tools ”的所有实例， 就可以为我们提供动态链接器的正确路径。还要调整specs文件，以便GCC知道在哪里可以找到正确的头文件和Glibc起始文件。一个sed的命令完成此：

```
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
```

目视检查规格文件以验证是否实际进行了更改是一个好主意。

此时必须确保调整后的工具链的基本功能（编译和链接）能够按预期工作。为此，请执行以下健全性检查：

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

应该没有错误，最后一个命令的输出将是（允许特定于平台的动态链接器名称有所不同）：

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

请注意，在64位系统上，/lib 是我们动态链接程序的位置，但是可以通过/ lib64中的符号链接进行访问。

> **注意**
>
> 在32位系统上，解释器应为/lib/ld-linux.so.2。

现在确保我们已设置为使用正确的启动文件：

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

最后一条命令的输出应为：

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

验证编译器正在搜索正确的头文件：

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

此命令应返回以下输出：

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

接下来，验证新链接程序是否与正确的搜索路径一起使用：

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

对带有'-linux-gnu'组件的路径的引用应被忽略，否则最后一条命令的输出应为：

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

接下来，确保我们使用的是正确的libc：

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

最后一条命令的输出应为：

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

最后，确保GCC使用了正确的动态链接器：

```
grep found dummy.log
```

最后一个命令的输出应为（允许特定于平台的动态链接器名称有所不同）：

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

如果输出没有如上所示出现或根本没有收到，则说明存在严重错误。调查并追溯步骤以找出问题所在并纠正。最可能的原因是规格文件调整出现问题。任何问题都需要解决，然后才能继续进行。

一切正常运行后，清理测试文件：

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

6.11。Zlib-1.2.11

Zlib软件包包含某些程序使用的压缩和解压缩例程。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	5.1 MB

6.11.1。安装Zlib

准备Zlib进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

共享库需要移到/lib，因此需要重新创建其中的.so文件/usr/lib：

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

6.11.2。Zlib的内容

已安装的库： libz。{a , so}

简短说明

libz	包含某些程序使用的压缩和解压缩功能

6.12。文件-5.37

文件包包含用于确定给定文件或多个文件类型的实用程序。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	19 MB

6.12.1。文件安装

准备要编译的文件：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.12.2。文件内容

安装的程序： 文件
已安装的库： libmagic.so

简短说明

文件	尝试对每个给定文件进行分类；它通过执行几个测试来做到这一点-文件系统测试，幻数测试和语言测试
libmagic	包含 文件 程序 使用的魔术数字识别例程

6.13。Readline-8.0

Readline软件包是一组提供命令行编辑和历史记录功能的库。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	15 MB

6.13.1。安装Readline

重新安装Readline将导致旧库移至<libraryname> .old。尽管通常这不是问题，但在某些情况下，它可能会触发ldconfig中的链接错误 。可以通过发出以下两个sed来避免这种情况：

```
sed -i '/MV.*old/d' Makefile.in
sed -i '/{OLDSUFF}/c:' support/shlib-install
```

准备Readline进行编译：

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

编译软件包：

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

make选项的含义：

SHLIB_LIBS="-L/tools/lib -lncursesw"
此选项强制Readline链接到 libncursesw库。

该软件包没有测试套件。

安装软件包：

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

现在，将动态库移动到更合适的位置，并修复一些权限和符号链接：

```
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
```

如果需要，请安装文档：

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

6.13.2。Readline的内容

已安装的库： libhistory.so和libreadline.so
安装目录： / usr / include / readline和/usr/share/doc/readline-8.0

简短说明

libhistory	提供一致的用户界面，以调用历史记录
libreadline	提供一组用于处理在程序的交互式会话中输入的文本的命令。

6.14。M4-1.4.18

M4软件包包含一个宏处理器。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	33 MB

6.14.1。安装M4

首先，进行glibc-2.28所需的一些修复：

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

准备M4进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.14.2。M4的内容

安装的程序： m4

简短说明

m4	复制给定文件，同时扩展它们包含的宏[这些宏是内置的或用户定义的，可以接受任意数量的参数。除了执行宏扩展之外， m4 还具有内置功能，包括命名文件，运行Unix命令，执行整数算术，操纵文本，递归等 。m4 程序既可以用作编译器的前端，也可以用作宏处理器本身。]

6.15。BC-2.1.3

Bc软件包包含任意精度的数字处理语言。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	2.8 MB

6.15.1。Bc的安装

准备Bc进行编译：

```
PREFIX=/usr CC=gcc CFLAGS="-std=c99" ./configure.sh -G -O3
```

配置选项的含义：

CC=gcc CFLAGS="-std=c99"
这些参数指定要使用的编译器和C标准。

-O3
指定要使用的优化。

-G
省略没有GNU bc才能运行的测试套件的某些部分。

编译软件包：

```
make
```

要测试BC，请运行：

```
make test
```

安装软件包：

```
make install
```

6.15.2。Bc的内容

安装的程序： bc和dc

简短说明

公元前	命令行计算器
直流电	反抛光命令行计算器

6.16。Binutils-2.32

Binutils程序包包含一个链接器，一个汇编器和其他用于处理目标文件的工具。

	预计编制
时间：	7.4 SBU
所需磁盘空间：	5.1 GB

6.16.1。安装Binutils

通过执行简单的测试，验证PTY在chroot环境中是否正常工作：

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

此命令应输出以下内容：

```
spawn ls
```

相反，如果输出包含以下消息，则表明该环境未设置为正确的PTY操作。在运行Binutils和GCC的测试套件之前，需要解决此问题：

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

现在，删除一项阻止测试运行到完成的测试：

```
sed -i '/@\tincremental_copy/d' gold/testsuite/Makefile.in
```

Binutils文档建议在专用的构建目录中构建Binutils：

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

准备Binutils进行编译：

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

配置参数的含义：

--enable-gold
构建黄金链接器，并将其安装为ld.gold（以及默认链接器）。

--enable-ld=default
构建原始的bdf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（在具有较小字长的主机上）。在64位系统上可能不需要，但没有危害。

--with-system-zlib
使用已安装的zlib库，而不要构建包含的版本。

编译软件包：

```
make tooldir=/usr
```

make参数的含义：

tooldir=/usr

通常，tooldir（可执行文件最终将位于的目录）设置为 `$(exec_prefix)/$(target_alias)`。例如，x86_64机器会将其扩展为 `/usr/x86_64-unknown-linux-gnu`。因为这是一个自定义系统，所以`/usr`不需要此特定于目标的目录。`$(exec_prefix)/$(target_alias)`如果系统用于交叉编译（例如，在Intel机器上编译软件包以生成可以在PowerPC机器上执行的代码），则将使用该命令。

> **重要**
>
> 本节中针对Binutils的测试套件被认为是至关重要的。在任何情况下都不要跳过它。

测试结果：

```
make -k check
```

在LFS环境中，PC相对偏移测试和debug_msg.sh测试可能会失败。

安装软件包：

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

6.16.2。Binutils的内容

安装的程序： addr2line，ar，as，c ++ filt，dwp，elfedit，gprof，ld，ld.bfd，ld.gold，nm，objcopy，objdump，ranlib，readelf，大小，字符串和strip

已安装的库： libbfd。{a，so}和libopcodes。{a，so}

安装目录： / usr / lib / ldscripts

简短说明

addr2line	将程序地址转换为文件名和行号；给定地址和可执行文件的名称，它将使用可执行文件中的调试信息来确定与该地址相关联的源文件和行号
AR	从档案创建，修改和提取
如	汇编程序，将 gcc 的输出汇编 成目标文件
C ++过滤	链接器使用它来分解C ++和Java符号并防止重载函数冲突
dwp	DWARF打包实用程序

elfedit	更新ELF文件的ELF标头
gprof	显示通话图配置文件数据
ld	链接器，它将多个对象和归档文件组合成一个文件，重新定位它们的数据并绑定符号引用
金	ld的简化版本，仅支持elf目标文件格式
ld.bfd	硬链接到 ld
纳米	列出给定目标文件中出现的符号
objcopy	将一种类型的目标文件转换为另一种类型
objdump	显示有关给定目标文件的信息，并带有控制要显示的特定信息的选项；显示的信息对使用编译工具的程序员很有用
兰利布	生成档案内容的索引并将其存储在档案中；索引列出了可重定位目标文件的归档成员定义的所有符号
Readelf	显示有关ELF类型二进制文件的信息
尺寸	列出给定目标文件的节大小和总大小
弦	对于每个给定的文件，输出至少可指定长度（默认为四个）的可打印字符序列；对于目标文件，默认情况下，它仅打印初始化和加载部分中的字符串，而对于其他类型的文件，它将扫描整个文件
跳闸	丢弃目标文件中的符号
libbfd	二进制文件描述符库
libopcodes	一个用于处理操作码的库- 处理器指令的 “ 可读文本 ”版本；它用于构建 objdump之类的 实用程序

6.17。GMP-6.1.2

GMP软件包包含数学库。它们具有用于任意精度算术的有用功能。

	预计编制
时间：	1.2 SBU
所需磁盘空间：	61 MB

6.17.1。安装GMP

注意

如果要针对32位x86进行构建，但是您具有可以运行64位代码的CPU，*并且*已CFLAGS在环境中指定 了配置脚本，则configure脚本将尝试为64位进行配置，但会失败。通过使用以下命令调用configure命令来避免这种情况

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

注意

GMP的默认设置会生成针对主机处理器优化的库。如果需要适合于能力不如主机CPU的处理器的库，则可以通过运行以下命令来创建通用库：

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

准备要编译的GMP：

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

新的配置选项的含义：

--enable-cxx
此参数启用C ++支持

--docdir=/usr/share/doc/gmp-6.1.2
此变量指定文档的正确位置。

编译软件包并生成HTML文档：

```
make
make html
```

重要

本节中的GMP测试套件被认为是至关重要的。在任何情况下都不要跳过它。

测试结果：

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

警告

gmp中的代码针对其所在的处理器进行了高度优化。有时，检测处理器的代码会误识别系统功能，并且使用带有消息"非法指令"的gmp库会在测试或其他应用程序中出现错误。在这种情况下，应使用选项--build = x86_64-unknown-linux-gnu重新配置gmp并重新构建。

确保测试套件中的所有190个测试均通过。通过发出以下命令来检查结果：

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

安装软件包及其文档：

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

6.17.2。GMP内容

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

简短说明

libgmp	包含精密数学函数
libgmpxx	包含C ++精度数学函数

6.18。MPFR-4.0.2

MPFR软件包包含用于多精度数学的函数。

	预计编制
时间：	0.9 SBU
所需磁盘空间：	37 MB

6.18.1。安装MPFR

准备要编译的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
```

> **重要**
>
> 本节中用于MPFR的测试套件被认为是关键的。在任何情况下都不要跳过它。

测试结果并确保所有测试均通过：

```
make check
```

安装软件包及其文档：

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

6.18.2。MPFR的内容

已安装的库： libmpfr.so
安装目录： /usr/share/doc/mpfr-4.0.2

简短说明

libmpfr	包含多精度数学函数

6.19。MPC-1.1.0

MPC软件包包含一个库，用于以任意高的精度对复数进行算术运算，并对结果进行正确的舍入。

	预计编制
时间：	0.3 SBU
所需磁盘空间：	22 MB

6.19.1。安装MPC

准备要编译的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
```

要测试结果，请发出：

```
make check
```

安装软件包及其文档：

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

6.19.2。MPC的内容

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

简短说明

libmpc	包含复杂的数学函数

6.20。暗影-4.7

Shadow软件包包含用于以安全方式处理密码的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	46 MB

6.20.1。影子的安装

注意

如果要强制使用强密码，请 在构建Shadow之前，先参考 **http://www.linuxfromscratch.org/blfs/view/9.0/postlfs/cracklib.html**来安装CrackLib。然后添加 *--with-libcrack*到 下面的 configure命令。

禁用安装group程序及其手册页，因为Coreutils提供了更好的版本。另外，禁止安装第6.8节“ Man-pages-5.02”中已经安装的手册页：

```
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 / /'    {} \;
```

可以使用更安全的*SHA-512* 密码加密方法来代替默认的 *crypt*方法，该方法还允许密码长度超过8个字符。还必须将 Shadow默认使用的用户邮箱的过时位置更改为当前使用的位置： /var/spool/mail/var/mail

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
       -e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

注意

如果选择在Cracklib支持下构建Shadow，请运行以下命令：

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' etc/login.defs
```

进行较小的更改以使useradd 1000生成的第一个组号：

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

准备Shadow进行编译：

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

configure选项的含义：

--with-group-name-max-length=32
最大用户名是32个字符。使最大组名相同。

编译软件包：

```
make
```

该软件包没有测试套件。

安装软件包：

```
make install
```

将放错位置的程序移到正确的位置：

```
mv -v /usr/bin/passwd /bin
```

6.20.2。配置阴影

该软件包包含用于添加，修改和删除用户和组的实用程序。设置和更改他们的密码；并执行其他管理任务。有关*密码屏蔽的*含义的完整说明 ，请参见doc/HOWTO见解压缩的源树中文件。如果使用Shadow支持，请记住需要验证密码的程序（显示管理器，FTP程序，pop3守护程序等）必须符合Shadow。也就是说，他们需要能够使用阴影密码。

要启用影子密码，请运行以下命令：

```
pwconv
```

要启用隐藏的组密码，请运行：

```
grpconv
```

Shadow的useradd实用工具的库存配置有一些警告，需要一些解释。首先，useradd 实用程序的默认操作是创建用户以及与该用户同名的组。默认情况下，用户ID（UID）和组ID（GID）编号将以1000开头。这意味着，如果不将参数传递给useradd，则每个用户将是系统上唯一组的成员。如果这种行为是不受欢迎的，则需要将*-g*参数传递给useradd。默认参数存储在/etc/default/useradd文件。您可能需要修改此文件中的两个参数以适合您的特定需求。

/etc/default/useradd **参数说明**

GROUP=1000

此参数设置/ etc / group文件中使用的组号的开头。您可以将其修改为所需的任何内容。请注意，useradd将永远不会重复使用UID或GID。如果使用了此参数中标识的数字，它将使用此后的下一个可用数字。还要注意，如果第一次使用 不带 参数的useradd时，系统上没有组1000 *-g*，则会在终端上显示以下消息：useradd: unknown GID 1000。您可以忽略此消息，并且将使用组号1000。

CREATE_MAIL_SPOOL=yes

此参数使useradd为新创建的用户创建邮箱文件。useradd将使该文件的组所有权成为*mail*具有0660权限的组。如果您希望这些邮箱文件不是由useradd创建的，请发出以下命令：

```
sed -i 's/yes/no/' /etc/default/useradd
```

6.20.3。设置root密码

为*root*用户选择一个密码，并通过运行以下命令进行设置：

```
passwd root
```

6.20.4。影子的内容

已安装程序： chage，chfn，chgpasswd，chpasswd，chsh，到期，faillog，gpasswd，groupadd，groupdel，groupmems，groupmod，grpck，grpconv，grpunconv，lastlog，login，logoutd，newgidmap，newgrp，newuidmap，newusers，nologin ，pwck，pwconv，pwunconv，sg（链接到newgrp），su，useradd，userdel，usermod，vigr（链接到vipw）和vipw

安装目录： / etc / default

简短说明

CHAGE	用于更改强制性密码更改之间的最大天数
chfn	用于更改用户的全名和其他信息
chgpasswd	用于以批处理方式更新组密码
密码	用于批量更新用户密码
chsh	用于更改用户的默认登录外壳
到期	检查并强制执行当前的密码过期策略
故障日志	用于检查登录失败的日志，设置帐户被阻止之前的最大失败次数或重置失败计数
密码	用于向组中添加和删除成员和管理员
组添加	用给定名称创建一个组
groupdel	删除具有给定名称的组
组成员	允许用户管理自己的组成员资格列表，而无需超级用户特权。
组mod	用于修改给定组的名称或GID
grpck	验证组文件的完整性， /etc/group并 /etc/gshadow
grpconv	从普通组文件创建或更新影子组文件
grpunconv	更新/etc/group 从/etc/gshadow然后删除后
最新记录	报告所有用户或给定用户的最新登录信息
登录	由系统用来让用户登录
注销	是用于强制限制登录时间和端口的守护程序

newgidmap	用于设置用户名称空间的gid映射
newgrp	用于在登录会话期间更改当前的GID
newuidmap	用于设置用户名称空间的uid映射
新用户	用于创建或更新整个系列的用户帐户
Nologin	显示一条消息，说明帐户不可用；它旨在用作已禁用帐户的默认外壳程序
密码	用于更改用户或组帐户的密码
w	验证密码文件的完整性， /etc/passwd并 /etc/shadow
pwconv	从普通密码文件创建或更新影子密码文件
普旺康夫	更新/etc/passwd 从/etc/shadow然后删除后
sg	当用户的GID设置为给定组的GID时执行给定命令
su	运行具有替代用户和组ID的Shell
用户添加	使用给定名称创建新用户，或更新默认的新用户信息
用户名	删除给定的用户帐户
用户模组	用于修改给定用户的登录名，用户标识（UID），shell，初始组，主目录等。
维格	编辑/etc/group 或/etc/gshadow文件
威宝	编辑/etc/passwd 或/etc/shadow文件

6.21。GCC-9.2.0

GCC软件包包含GNU编译器集合，其中包括C和C ++编译器。

	预计编制
时间：	95 SBU（带测试）
所需磁盘空间：	4.2 GB

6.21.1。安装GCC

如果基于x86_64构建，请将64位库的默认目录名称更改为“ lib ”：

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

GCC文档建议在专用的构建目录中构建GCC：

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

准备要编译的GCC：

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

请注意，对于其他语言，有一些前提条件尚不可用。有关如何构建GCC支持的所有语言的说明，请参阅 **BLFS手册**。

新的configure参数的含义：

SED=sed
设置此环境变量可防止对/ tools / bin / sed进行硬编码的路径。

--with-system-zlib
此开关告诉GCC链接到系统安装的Zlib库副本，而不是其自己的内部副本。

编译软件包：

```
make
```

重要

在本节中，GCC测试套件被认为是至关重要的。在任何情况下都不要跳过它。

已知GCC测试套件中的一组测试会耗尽堆栈，因此在运行测试之前增加堆栈大小：

```
ulimit -s 32768
```

以非特权用户身份测试结果，但不要因错误而停止：

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

要获得测试套件结果的摘要，请运行：

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

仅对于摘要，将输出通过管道传输 grep -A7 Summ。

可以将结果与位于**http://www.linuxfromscratch.org/lfs/build-logs/9.0/** 和**https://gcc.gnu.org/ml/gcc-testresults/**的结果进行比较。

已知与get_time相关的六个测试失败。这些显然与en_HK语言环境有关。

已知在LFS chroot环境中，两个名为实验/ net的名为lookup.cc和reverse.cc的测试在LFS chroot环境中失败，因为它们需要/ etc / hosts和iana-etc。

已知两个名为pr57193.c和pr90178.c的测试失败。

某些无法预料的故障总是无法避免的。GCC开发人员通常知道这些问题，但尚未解决。除非测试结果与上述URL的测试结果有很大不同，否则可以安全地继续。

安装软件包并删除不需要的目录：

```
make install
rm -rf /usr/lib/gcc/$(gcc -dumpmachine)/9.2.0/include-fixed/bits/
```

GCC构建目录*nobody*现在已经拥有，并且已安装的标头目录（及其内容）的所有权将不正确。将所有权更改为*root*用户和组：

```
chown -v -R root:root \
    /usr/lib/gcc/*linux-gnu/9.2.0/include{,-fixed}
```

创建 **FHS**出于"历史"原因所需的符号链接。

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

许多软件包使用名称cc来调用C编译器。要满足这些软件包，请创建一个符号链接：

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

添加一个兼容性符号链接，以启用带有链接时间优化（LTO）的程序：

```
install -v -dm755 /usr/lib/bfd-plugins
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/9.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

现在我们已经有了最终的工具链，再次确保编译和链接将按预期工作将很重要。为此，我们将执行与本章前面部分相同的健全性检查：

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

应该没有错误，最后一个命令的输出将是（允许特定于平台的动态链接器名称有所不同）：

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

现在确保我们已设置为使用正确的启动文件：

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

最后一条命令的输出应为：

```
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/../../../../lib/crtn.o succeeded
```

取决于您的计算机体系结构，以上内容可能会略有不同，差异通常是之后的目录名称/usr/lib/gcc。在这里要查找的重要内容是gcccrt*.o在/usr/lib目录下找到了所有三个文件 。

验证编译器正在搜索正确的头文件：

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

此命令应返回以下输出：

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/9.2.0/include-fixed
 /usr/include
```

同样，请注意，根据您的体系结构，以目标三元组命名的目录可能与上面的目录不同。

接下来，验证新链接程序是否与正确的搜索路径一起使用：

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

对带有'-linux-gnu'组件的路径的引用应被忽略，否则最后一条命令的输出应为：

```
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位系统可能会看到几个不同的目录。例如，以下是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");
```

接下来，确保我们使用的是正确的libc：

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

最后一条命令的输出应为：

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

最后，确保GCC使用了正确的动态链接器：

```
grep found dummy.log
```

最后一个命令的输出应为（允许特定于平台的动态链接器名称有所不同）：

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

如果输出没有如上所示出现或根本没有收到，则说明存在严重错误。调查并追溯步骤以找出问题所在并纠正。最可能的原因是规格文件调整出现问题。任何问题都需要解决，然后才能继续进行。

一切正常运行后，清理测试文件：

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

最后，移动放错位置的文件：

```
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的内容

安装的程序： c ++，cc（链接到gcc），cpp，g ++，gcc，gcc-ar，gcc-nm，gcc-ranlib，gcov，gcov-dump和gcov-tool

已安装的库： libasan。{a，so}，libatomic。{a，so}，libcc1.so，libgcc.a，libgcc_eh.a，libgcc_s.so，libgcov.a，libgomp。{a，so}，libitm。{a，so}，liblsan。{a，so}，liblto_plugin.so，libquadmath。{a，so}，libssp。{a，so}，libssp_nonshared.a，libstdc ++。{a，so}，libstdc ++ fs。 a，libsupc ++。a，libtsan。{a，so}和libubsan。{a，so}

安装目录： / usr / include / c ++，/ usr / lib / gcc，/ usr / libexec / gcc和/usr/share/gcc-9.2.0

简短说明

C ++	C ++编译器
抄送	C编译器
cpp	C预处理程序；编译器使用它来扩展源文件中的＃include，＃define和类似语句
g ++	C ++编译器
海湾合作委员会	C编译器
海湾合作委员会	ar 周围的包装器，将插件添加到命令行。该程序仅用于添加“链接时间优化”，对于默认构建选项不起作用
碳纳米管	nm 周围的包装器，将插件添加到命令行。该程序仅用于添加“链接时间优化”，对于默认构建选项不起作用
gcc-ranlib	围绕 ranlib 的包装器，将插件添加到命令行。该程序仅用于添加“链接时间优化”，对于默认构建选项不起作用
gcov	覆盖率测试工具；它用于分析程序，以确定最有效的地方
gcov转储	离线gcda和gcno配置文件转储工具
gcov工具	离线gcda配置文件处理工具
libasan	Address Sanitizer运行时库
libatomic	GCC原子内置运行时库
libcc1	C预处理库
libgcc	包含对 gcc的 运行时支持
libgcov	当指示GCC启用性能分析时，此库链接到程序
libgomp	OpenMP API的GNU实现，用于C / C ++和Fortran中的多平台共享内存并行编程
liblsan	泄漏消毒剂运行时库
liblto_plugin	GCC的链接时间优化（LTO）插件使GCC可以跨编译单元执行优化

libquadmath	GCC Quad Precision数学库API
libssp	包含支持GCC堆栈破坏保护功能的例程
libstdc++	标准的C ++库
libstdc++fs	ISO / IEC TS 18822：2015文件系统库
libsupc++	提供C ++编程语言的支持例程
libtsan	线程清理程序运行时库
libubsan	Undefined Behavior Sanitizer运行时库

6.22。Bzip2-1.0.8

Bzip2软件包包含用于压缩和解压缩文件的程序。与传统的gzip相比，使用bzip2压缩文本文件产生的压缩百分比要好得多。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	7.7 MB

6.22.1。安装Bzip2

应用补丁程序将安装此软件包的文档：

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

以下命令确保符号链接的安装是相对的：

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

确保手册页安装在正确的位置：

```
sed -i "s@(PREFIX)/man@(PREFIX)/share/man@g" Makefile
```

准备使用以下命令进行编译的Bzip2：

```
make -f Makefile-libbz2_so
make clean
```

make参数的含义：

-f Makefile-libbz2_so

这将导致使用其他Makefile文件（在本例中为Makefile-libbz2_so文件）构建Bzip2， 该文件将创建动态libbz2.so库并将Bzip2实用程序链接到该库。

编译并测试软件包：

```
make
```

安装程序：

```
make PREFIX=/usr install
```

将共享的bzip2二进制文件安装到 /bin目录中，进行一些必要的符号链接，然后清理：

```
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的内容

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bzmore）和bzmore

已安装的库： libbz2。{a，so}

安装目录： /usr/share/doc/bzip2-1.0.8

简短说明

bunzip2	解压缩压缩文件
猫咪	解压缩到标准输出
bzcmp	在bzip压缩文件上 运行 cmp
bzdiff	在bzip压缩文件上 运行 diff
Bzegrep	在压缩文件上 运行 egrep
bzfgrep	在压缩文件上 运行 fgrep
bzgrep	在压缩文件上 运行 grep
bzip2	使用具有霍夫曼编码的Burrows-Wheeler块排序文本压缩算法压缩文件；压缩率比使用“ Lempel-Ziv ”算法（例如 gzip）的 更传统的压缩机要好。
bzip2recover	尝试从损坏的压缩文件中恢复数据

bzless	在压缩文件上 运行 更少
bzmore	在bzip压缩文件上 运行 更多
libbz2	该库使用Burrows-Wheeler算法实现无损块分类数据压缩

6.23。包配置0.29.2

pkg-config软件包包含一个工具，用于在配置和执行文件期间传递包含路径和/或库路径以构建工具。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	30 MB

6.23.1。安装Pkg-config

准备Pkg-config进行编译：

```
./configure --prefix=/usr              \
            --with-internal-glib       \
            --disable-host-tool        \
            --docdir=/usr/share/doc/pkg-config-0.29.2
```

新的配置选项的含义：

--with-internal-glib
这将允许pkg-config使用其内部版本的Glib，因为LFS中不提供外部版本。

--disable-host-tool
此选项禁止创建到pkg-config程序的不需要的硬链接。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.23.2。Pkg-config的内容

安装的程序： pkg-config
安装目录： /usr/share/doc/pkg-config-0.29.2

简短说明

pkg-config	返回指定库或包的元信息

6.24。Ncurses-6.1

Ncurses软件包包含用于终端独立处理字符屏幕的库。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	42 MB

6.24.1。Ncurses的安装

不要安装未由configure处理的静态库：

```
sed -i '/LIBTOOL_INSTALL/d' c++/Makefile.in
```

准备Ncurses进行编译：

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --enable-pc-files       \
            --enable-widec
```

新的配置选项的含义：

--enable-widec
此切换导致libncursesw.so.6.1构建宽字符库（例如）而不是普通库（例如 libncurses.so.6.1）。这些宽字符库可用于多字节和传统的8位语言环境，而普通库仅可在8位语言环境中正常工作。宽字符库和普通库是源兼容的，但不是二进制兼容的。

--enable-pc-files
此开关为pkg-config生成并安装.pc文件。

--without-normal

此开关禁用构建和安装大多数静态库。

编译软件包：

```
make
```

该软件包具有一个测试套件，但是只能在安装了该软件包之后才能运行。测试位于 test/目录中。有关README更多详细信息，请参见该目录中的 文件。

安装软件包：

```
make install
```

将共享库移动到/lib预期驻留的目录：

```
mv -v /usr/lib/libncursesw.so.6* /lib
```

因为库已被移动，所以一个符号链接指向一个不存在的文件。重新创建它：

```
ln -sfv ../../lib/$(readlink /usr/lib/libncursesw.so) /usr/lib/libncursesw.so
```

许多应用程序仍然期望链接器能够找到非宽字符的Ncurses库。通过符号链接和链接描述文件，诱使此类应用程序与宽字符库链接：

```
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-lcurses在构建时查找的旧应用程序 仍可构建：

```
rm -vf                     /usr/lib/libcursesw.so
echo "INPUT(-lncursesw)" > /usr/lib/libcursesw.so
ln -sfv libncurses.so      /usr/lib/libcurses.so
```

如果需要，请安装Ncurses文档：

```
mkdir -v       /usr/share/doc/ncurses-6.1
cp -v -R doc/* /usr/share/doc/ncurses-6.1
```

注意

上面的说明不会创建非宽字符的Ncurses库，因为从源代码编译安装的任何软件包都不会在运行时链接到它们。但是，链接到非宽字符Ncurses库的唯一已知的仅二进制应用程序需要版本5。如果由于某些仅二进制应用程序或与LSB兼容而必须具有此类库，请使用以下代码再次构建软件包命令：

```
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的内容

安装的程序： captoinfo（链接到tic），clear，infocmp，infotocap（链接到tic），ncursesw6-config，reset（链接到tset），制表符，tic，toe，tput和tset

已安装的库： libcursesw.so（指向libncursesw.so的符号链接和链接脚本），libformw.so，libmenuw.so，libncursesw.so，libncurses ++ wa，libpanelw.so及其不含“ w”的非宽字符副本在库名称中。

安装目录： / usr / share / tabset，/ usr / share / terminfo和/usr/share/doc/ncurses-6.1

简短说明

captoinfo	将termcap描述转换为terminfo描述
明确	清除屏幕（如果可能）
infocmp	比较或打印terminfo描述
infotocap	将terminfo描述转换为termcap描述
ncursesw6-config	提供ncurses的配置信息
重启	将终端重新初始化为其默认值
标签	清除并设置终端上的制表位
抽动	terminfo条目描述编译器，将terminfo文件从源格式转换为ncurses库例程所需的二进制格式[terminfo文件包含有关特定终端功能的信息。
脚趾	列出所有可用的终端类型，并给出每种的主要名称和描述

投入	使依赖于终端的功能的值可用于外壳程序；它也可以用于重置或初始化终端或报告其长名
预设	可用于初始化终端
libcursesw	链接到 libncursesw
libncursesw	包含在终端屏幕上以多种复杂方式显示文本的功能；使用这些功能的一个很好的例子是内核的 make menuconfig 期间显示的菜单。
libformw	包含实现表格的功能
libmenuw	包含实现菜单的功能
libpanelw	包含实现面板的功能

6.25。Attr-2.4.48

attr软件包包含实用程序，用于管理文件系统对象上的扩展属性。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	4.2 MB

6.25.1。安装Attr

准备要编译的Attr：

```
./configure --prefix=/usr     \
            --bindir=/bin     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.4.48
```

编译软件包：

```
make
```

这些测试需要在支持扩展属性（例如ext2，ext3或ext4文件系统）的文件系统上运行。要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

共享库需要移到/lib，因此需要重新创建其中的.so文件/usr/lib：

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

6.25.2。Attr的内容

已安装程序： attr，getfattr和setfattr
已安装的库： libattr.so
安装目录： / usr / include / attr和/usr/share/doc/attr-2.4.48

简短说明

属性	扩展文件系统对象的属性
getfattr	获取文件系统对象的扩展属性
setfattr	设置文件系统对象的扩展属性
libattr	包含用于处理扩展属性的库函数

6.26。ACL-2.2.53

Acl程序包包含用于管理访问控制列表的实用程序，这些实用程序用于为文件和目录定义更细粒度的自由访问权限。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	6.4 MB

6.26.1。安装Acl

准备要编译的Acl：

```
./configure --prefix=/usr         \
            --bindir=/bin         \
            --disable-static      \
            --libexecdir=/usr/lib \
            --docdir=/usr/share/doc/acl-2.2.53
```

编译软件包：

```
make
```

在使用Acl库构建Coreutils之后，需要在支持访问控制的文件系统上运行Acl测试。如果需要，请 在本章稍后构建Coreutils之后返回此软件包并运行 `make check`。

安装软件包：

```
make install
```

共享库需要移到`/lib`，因此需要重新创建其中的`.so`文件`/usr/lib`：

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

6.26.2。Acl的内容

已安装程序： 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

简短说明

查克尔	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	包含用于处理访问控制列表的库函数

6.27。Libcap 2.27

Libcap软件包实现了Linux内核中可用的POSIX 1003.1e功能的用户空间接口。这些功能是将所有强大的root特权划分为一组不同的特权。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	1.5 MB

6.27.1。Libcap的安装

防止安装静态库：

```
sed -i '/install.*STALIBNAME/d' libcap/Makefile
```

编译软件包：

```
make
```

该软件包没有测试套件。

安装软件包：

```
make RAISE_SETFCAP=no lib=lib prefix=/usr install
chmod -v 755 /usr/lib/libcap.so.2.27
```

make选项的含义：

RAISE_SETFCAP=no
此参数跳过尝试对其自身使用setcap的尝试。如果内核或文件系统不支持扩展功能，则可以避免安装错误。

lib=lib
该参数将库安装在$prefix/lib而不是$prefix/lib64x86_64上。它对x86没有影响。

共享库需要移到/lib，因此需要重新创建其中的.so文件/usr/lib：

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

6.27.2。Libcap的内容

安装的程序： capsh，getcap，getpcaps和setcap
已安装的库： libcap.so

简短说明

卡什	一个外壳程序，以探索和限制功能支持
getcap	检查文件功能
getpcaps	显示查询过程中的功能
设定值	设置文件功能
libcap	包含用于操纵POSIX 1003.1e功能的库函数

6.28。Sed-4.7

Sed程序包包含一个流编辑器。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	32 MB

6.28.1。安装Sed

首先解决LFS环境中的问题并删除失败的测试：

```
sed -i 's/usr/tools/'                 build-aux/help2man
sed -i 's/testsuite.panic-tests.sh//' Makefile.in
```

准备Sed进行编译：

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

编译软件包并生成HTML文档：

```
make
make html
```

要测试结果，请发出：

```
make check
```

安装软件包及其文档：

```
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的内容

安装的程序：	sed
安装目录：	/usr/share/doc/sed-4.7

简短说明

sed	一次过滤和转换文本文件

6.29。Psmisc-23.2

Psmisc程序包包含用于显示有关正在运行的进程的信息的程序。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	4.6 MB

6.29.1。安装Psmisc

准备Psmisc进行编译：

```
./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killall和fuser程序移至FHS指定的位置：

```
mv -v /usr/bin/fuser   /bin
mv -v /usr/bin/killall /bin
```

6.29.2。Psmisc的内容

安装的程序： fuser，killall，peekfd，prtstat，pslog，pstree和pstree.x11（链接到pstree）

简短说明

热熔器	报告使用给定文件或文件系统的进程的进程ID（PID）
杀死所有	按名称杀死进程；它向运行任何给定命令的所有进程发送信号

偷看	鉴于其PID，窥视正在运行的进程的文件描述符
prtstat	打印有关流程的信息
pslog	报告进程的当前日志路径
pstree	将正在运行的进程显示为树
pstree.x11	与 pstree 相同，除了它在退出前等待确认

6.30。IANA-ETC-2.30

Iana-Etc程序包为网络服务和协议提供数据。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	2.3 MB

6.30.1。安装Iana-Etc

以下命令将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的内容

安装的文件： / etc / protocols和/ etc / services

简短说明

/etc/protocols	描述可从TCP / IP子系统获得的各种DARPA Internet协议
/etc/services	提供Internet服务的友好文本名称与其基础分配的端口号和协议类型之间的映射

6.31。野牛3.4.1

Bison软件包包含一个解析器生成器。

	预计编制
时间：	0.3 SBU
所需磁盘空间：	39 MB

6.31.1。野牛的安装

首先，解决当前版本的构建问题：

```
sed -i '6855 s/mv/cp/' Makefile.in
```

准备Bison进行编译：

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.4.1
```

编译该软件包，但在当前版本中解决争用条件：

```
make -j1
```

在检查方面，野牛和flex之间存在循环依赖关系。如果需要，在下一部分中安装flex之后，可以重建bison软件包，并可以使用`make check`运行bison检查。

安装软件包：

```
make install
```

6.31.2。野牛的内容

已安装程序：	bison和yacc
已安装的库：	liby.a
安装目录：	/ usr / share / bison

简短说明

野牛	根据一系列规则生成用于分析文本文件结构的程序；Bison替代了Yacc（又是另一个编译器）
yacc	野牛 包装，用于仍然调用 yacc 而不是 野牛的程序 ；它 用 选项 称 野牛 *-y*
liby	Yacc库包含Yacc兼容`yyerror`和`main`函数的实现；这个库通常不是很有用，但是POSIX需要它

6.32。Flex-2.6.4

Flex程序包包含一个实用程序，用于生成识别文本模式的程序。

	预计编制
时间：	0.5 SBU
所需磁盘空间：	36 MB

6.32.1。安装Flex

首先，解决glibc-2.26引入的问题：

```
sed -i "/math.h/a #include <malloc.h>" src/flexdef.h
```

生成过程假定help2man程序可用于从可执行文件--help选项创建手册页。这不存在，因此我们使用环境变量来跳过此过程。现在，准备Flex进行编译：

```
HELP2MAN=/tools/bin/true \
./configure --prefix=/usr --docdir=/usr/share/doc/flex-2.6.4
```

编译软件包：

```
make
```

要测试结果（大约0.5 SBU），请发出：

```
make check
```

安装软件包：

```
make install
```

一些程序尚不了解flex并尝试运行其前身lex。为了支持这些程序，请创建一个以lex仿真模式lex运行的符号链接： flex

```
ln -sv flex /usr/bin/lex
```

6.32.2。Flex的内容

安装的程序：	flex，flex ++（链接到flex）和lex（链接到flex）
已安装的库：	libfl.so
安装目录：	/usr/share/doc/flex-2.6.4

简短说明

柔性	用于生成识别文本模式的程序的工具；它允许通用性指定模式查找规则，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。这是 flex 的符号链接
莱克斯	在 lex 仿真模式下 运行 flex的 符号链接
libfl	该flex库

6.33。Grep-3.3

Grep软件包包含用于搜索文件的程序。

	预计编制
时间：	0.5 SBU
所需磁盘空间：	37 MB

6.33.1。安装Grep

准备Grep进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make -k check
```

安装软件包：

```
make install
```

6.33.2。Grep的内容

已安装程序： egrep，fgrep和grep

简短说明

egrep	打印与扩展的正则表达式匹配的行
fgrep	打印与固定字符串列表匹配的行
grep	打印与基本正则表达式匹配的行

6.34。Bash-5.0

Bash软件包包含Bourne-Again SHell。

	预计编制
时间：	2.1 SBU
所需磁盘空间：	62 MB

6.34.1。Bash的安装

准备Bash进行编译：

```
./configure --prefix=/usr                     \
            --docdir=/usr/share/doc/bash-5.0 \
            --without-bash-malloc             \
            --with-installed-readline
```

新的configure选项的含义：

--with-installed-readline
此选项告诉Bash使用readline系统上已经安装的库，而不是使用其自己的readline版本。

编译软件包：

```
make
```

如果未运行测试套件，请 跳至“ 安装软件包 ”。

要准备测试，请确保*nobody*用户可以写入源代码树：

```
chown -Rv nobody .
```

现在，以*nobody*用户身份运行测试：

```
su nobody -s /bin/bash -c "PATH=$PATH HOME=/home make tests"
```

安装软件包并将主要可执行文件移至 /bin：

```
make install
mv -vf /usr/bin/bash /bin
```

运行新编译的bash程序（替换当前正在执行的程序）：

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

> **注意**
>
> 使用的参数使bash进程成为交互式登录外壳，并继续禁用哈希，以便在新程序可用时找到它们。

6.34.2。Bash的内容

已安装程序： bash，bashbug和sh（链接到bash）
安装目录： / usr / include / bash，/ usr / lib / bash和/usr/share/doc/bash-5.0

简短说明

重击	广泛使用的命令解释器；在执行之前，它会在给定的命令行上执行多种类型的扩展和替换，因此使此解释器成为强大的工具
ash	一个shell脚本，可帮助用户编写和邮寄有关 bash的 标准格式的错误报告
SH	bash 程序的 符号链接；当作为 sh 调用时， bash 尝试尽可能地模仿 sh 的历史版本的启动行为 ，同时也符合POSIX标准

6.35。Libtool 2.4.6

Libtool软件包包含GNU通用库支持脚本。它包装了在一致的可移植界面中使用共享库的复杂性。

	预计编制
时间：	1.9 SBU
所需磁盘空间：	43 MB

6.35.1。安装Libtool

准备要编译的Libtool：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

> **注意**
>
> 在具有多个内核的系统上，libtool的测试时间可以大大减少。为此，请将TESTSUITEFLAGS = -j <N>附加 到上面的行。例如，使用-j4可以将测试时间减少60%以上。

由于循环依赖，已知有五个测试在LFS构建环境中失败，但是如果在安装了automake之后重新检查，则所有测试都会通过。

安装软件包：

```
make install
```

6.35.2。Libtool的内容

安装的程序： libtool和libtoolize
已安装的库： libltdl.so
安装目录： / usr / include / libltdl和/ usr / share / libtool

简短说明

的libtool	提供广义的图书馆建设支持服务
libtoolize	提供将 libtool 支持添加到软件包 的标准方法
libltdl	隐藏了倾斜库的各种困难

6.36。GDBM-1.18.1

GDBM软件包包含GNU数据库管理器。它是一个数据库函数库，该库使用可扩展的散列，并且类似于标准的UNIX dbm。该库提供了用于存储键/数据对，通过其键搜索和检索数据以及删除键及其数据的原语。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	11 MB

6.36.1。安装GDBM

准备要编译的GDBM：

```
./configure --prefix=/usr    \
            --disable-static \
            --enable-libgdbm-compat
```

configure选项的含义：

--enable-libgdbm-compat

此开关使libgdbm兼容性库得以构建，因为LFS之外的某些软件包可能需要它提供的旧DBM例程。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.36.2。GDBM的内容

已安装的程序： gdbm_dump，gdbm_load和gdbmtool

已安装的库： libgdbm.so和libgdbm_compat.so

简短说明

gdbm_dump	将GDBM数据库转储到文件
	从转储文件重新创建GDBM数据库

gdbm_load	
gdbmtool	测试和修改GDBM数据库
libgdbm	包含操作散列数据库的函数
libgdbm_compat	包含旧DBM功能的兼容性库

6.37。Gperf-3.1

Gperf从密钥集生成完美的哈希函数。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	6.3 MB

6.37.1。安装Gperf

准备Gperf进行编译：

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

编译软件包：

```
make
```

如果运行多个同时测试（-j选项大于1），则测试会失败。要测试结果，请发出：

```
make -j1 check
```

安装软件包：

```
make install
```

6.37.2。Gperf的内容

安装的程序：	gperf
安装目录：	/usr/share/doc/gperf-3.1

简短说明

per	从密钥集中生成完美的哈希

6.38。外籍人士2.2.7

Expat软件包包含一个面向流的C库，用于解析XML。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	11 MB

6.38.1。安装Expat

首先解决LFS环境中的回归测试问题：

```
sed -i 's|usr/bin/env |bin/|' run.sh.in
```

准备要编译的Expat：

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.2.7
```

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

如果需要，请安装文档：

```
install -v -m644 doc/*.{html,png,css} /usr/share/doc/expat-2.2.7
```

6.38.2。外籍人士的内容

安装的程序： xmlwf
已安装的库： libexpat.so
安装目录： /usr/share/doc/expat-2.2.7

简短说明

xmlwf	是一种非验证实用程序，用于检查XML文档是否格式正确
libexpat	包含用于解析XML的API函数

6.39。的Inetutils-1.9.4

Inetutils软件包包含用于基本联网的程序。

	预计编制
时间：	0.3 SBU
所需磁盘空间：	29 MB

6.39.1。安装Inetutils

准备要编译的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
此选项可防止Inetutils安装记录 程序，脚本可使用该 记录程序将消息传递到系统日志守护程序。不要安装它，因为Util-linux安装了最新版本。

--disable-whois
此选项禁用了 过时的Inetutils whois客户端的构建 。 BLFS 手册中提供了有关更好的Whois客户的说明 。

*--disable-r**
这些参数禁用由于安全问题而不应使用的过时程序的构建。这些程序提供的功能可以通过BLFS手册中的openssh包提供。

--disable-servers

这将禁用安装作为Inetutils软件包一部分的各种网络服务器。这些服务器被认为在基本的LFS系统中不合适。有些是天生不安全的，仅在受信任的网络上才被认为是安全的。请注意，这些服务器中的许多服务器都有更好的替代品。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

注意

一个测试libls.sh在初始chroot环境中可能会失败，但是如果在LFS系统完成后重新运行该测试，则测试将通过。如果主机系统不具有ipv6功能，则一个测试ping-localhost.sh将失败。

安装软件包：

```
make install
```

移动一些程序，以便/usr在无法访问时可用：

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

6.39.2。Inetutils的内容

安装的程序： dnsdomainname，ftp，ifconfig，主机名，ping，ping6，talk，telnet，tftp和traceroute

简短说明

dns域名	显示系统的DNS域名
ftp	是文件传输协议程序
主机名	报告或设置主机名
ifconfig	管理网络接口
ping	发送回声请求数据包并报告答复需要多长时间

ping6	用于IPv6网络 的 ping 版本
谈论	用于与其他用户聊天
远程登录	TELNET协议的接口
ftp	一个简单的文件传输程序
跟踪路由	跟踪数据包从您正在使用的主机到网络上另一主机的路由，显示沿途的所有中间跃点（网关）

6.40。Perl-5.30.0

Perl软件包包含实用的提取和报告语言。

	预计编制
时间：	9.9 SBU
所需磁盘空间：	272 MB

6.40.1。安装Perl

首先创建一个/etc/hosts 要在Perl的配置文件之一以及可选的测试套件中引用的基本文件：

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

现在，此版本的Perl构建了Compress :: Raw :: Zlib和Compress :: Raw :: BZip2模块。默认情况下，Perl将使用源的内部副本进行构建。发出以下命令，以便Perl将使用系统上安装的库：

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

要完全控制Perl的设置方式，可以从以下命令中删除“ -des ”选项，然后手动选择此软件包的构建方式。或者，完全按照以下命令使用Perl自动检测的默认值：

```
sh Configure -des -Dprefix=/usr                \
                  -Dvendorprefix=/usr          \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"  \
                  -Duseshrplib                 \
                  -Dusethreads
```

配置选项的含义：

-Dvendorprefix=/usr

这样可以确保perl知道如何告诉软件包应该在哪里安装perl模块。

-Dpager="/usr/bin/less -isR"

这样可确保less使用而不是more。

-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3

由于尚未安装Groff，Configure认为我们不想要Perl的手册页。发出这些参数将覆盖此决定。

-Duseshrplib

构建一些perl模块所需的共享libperl。

-Dusethreads

使用线程支持构建perl。

编译软件包：

```
make
```

要测试结果（大约11 SBU），请发出：

```
make -k test
```

> **注意**
>
> 由于使用最新版本的gdbm，因此一项测试失败。

安装软件包并清理：

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

6.40.2。Perl的内容

已安装程序： corelist，cpan，enc2xs，encguess，h2ph，h2xs，instmodsh，json_pp，libnetcfg，perl，perl5.30.0（与perl的硬链接），perlbug，perldoc，perlivp，perlthanks（与perlbug的硬链接），piconv，pl2pm ，pod2html，pod2man，pod2text，pod2usage，podchecker，podselect，证明，ptar，ptardiff，ptargrep，shasum，splain，xsubpp和zipdetails

已安装的库： 许多不能在此处列出的库

安装目录： / usr / lib / perl5

简短说明

核心清单	到Module :: CoreList的命令行前端
pan	从命令行与综合Perl存档网络（CPAN）进行交互
enc2xs	从Unicode字符映射或Tcl编码文件为Encode模块构建Perl扩展
激怒	猜测一个或几个文件的编码类型
h2ph	将.hC头文件转换为.phPerl头文件
h2xs	将.hC头文件转换为Perl扩展名
安装	用于检查已安装的Perl模块的Shell脚本，并且可以从已安装的模块创建tarball
json_pp	在某些输入和输出格式之间转换数据
libnetcfg	可用于配置libnetPerl模块
佩尔	将C， sed ， awk 和 sh的 某些最佳功能组合 成一种单一的瑞士军语
perl5.30.0	与 perl 的硬链接
臭虫	用于生成有关Perl或它随附的模块的错误报告，并将其邮寄
Perldoc	以pod格式显示一份文档，该文档嵌入在Perl安装树或Perl脚本中
Perlivp	Perl安装验证过程；它可以用来验证Perl及其库是否已正确安装
谢谢	用于生成感谢消息以邮寄给Perl开发人员
微微	Perl版本的字符编码转换器 iconv
下午2点	用于将Perl4 .pl文件转换为Perl5 .pm模块的 粗略工具
pod2html	将文件从pod格式转换为HTML格式
pod2man	将pod数据转换为格式化的* roff输入
pod2text	将pod数据转换为格式化的ASCII文本
pod2用法	从文件中的嵌入式Pod文档中打印使用情况消息
podchecker	检查Pod格式文档文件的语法
荚选择	显示窗格文档的选定部分
证明	用于针对Test :: Harness模块运行测试的命令行工具
tar	一个 焦油 用Perl编写的程序样
帕塔迪夫	一种Perl程序，将提取的档案与未提取的档案进行比较
ptargrep	一个Perl程序，将模式匹配应用于tar归档文件中的文件内容
夏苏姆	打印或检查SHA校验和
pla	用于在Perl中强制进行详细的警告诊断
订阅	将Perl XS代码转换为C代码

拉链细节	显示有关Zip文件内部结构的详细信息

6.41。XML ::解析器2.44

XML :: Parser模块是James Clark的XML解析器Expat的Perl接口。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	2.3 MB

6.41.1。安装XML :: Parser

准备XML :: Parser进行编译：

```
perl Makefile.PL
```

编译软件包：

```
make
```

要测试结果，请发出：

```
make test
```

安装软件包：

```
make install
```

6.41.2。XML :: Parser的内容

已安装模块： Expat.so

简短说明

Expat	提供Perl Expat接口

6.42。Intltool-0.51.0

Intltool是一种国际化工具，用于从源文件中提取可翻译字符串。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	1.5 MB

6.42.1。安装Intltool

首先修复由perl-5.22和更高版本引起的警告：

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

准备要编译的Intltool：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
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的内容

已安装程序：	intltool-extract，intltool-merge，intltool-prepare，intltool-update和intltoolize
安装目录：	/usr/share/doc/intltool-0.51.0和/ usr / share / intltool

简短说明

信息化	准备一个软件包以使用intltool
提取工具	生成可由 gettext 读取的头文件
国际合并	将翻译后的字符串合并为各种文件类型

准备intltool	更新pot文件并将其与翻译文件合并
国际工具更新	更新po模板文件并将其与翻译合并

6.43。自动配置2.69

Autoconf软件包包含用于生成可自动配置源代码的Shell脚本的程序。

大概	编译
时间：	少于0.1 SBU（测试时约为3.4 SBU）
所需磁盘空间：	79 MB

6.43.1。安装Autoconf

首先，修复Perl 5.28生成的错误。

```
sed '361 s/{/\\{/' -i bin/autoscan.in
```

准备Autoconf进行编译：

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

编译软件包：

```
make
```

目前，bash-5和libtool-2.4.3破坏了该测试套件。无论如何要运行测试，请发出：

```
make check
```

安装软件包：

```
make install
```

6.43.2。Autoconf的内容

安装的程序：	autoconf，autoheader，autom4te，autoreconf，autoscan，autoupdate和ifnames
安装目录：	/ usr / share / autoconf

简短说明

自动配置	产生可自动配置软件源代码包以适应多种类Unix系统的shell脚本；它生成的配置脚本是独立的一运行它们不需要 autoconf 程序
自动标题	用于创建C *#define* 语句的模板文件以 供配置以使用的工具
自动化	M4宏处理器的包装
自动重新配置	以正确的顺序 自动运行 autoconf ， autoheader ， aclocal ， automake ， gettextize 和 libtoolize ，以节省更改 autoconf 和 automake 模板文件时的时间
自动扫描	帮助创建configure.in软件包文件；它检查目录树中的源文件，搜索它们中常见的可移植性问题，并创建一个configure.scan文件作为configure.in该软件包的初步文件
自动更新	修改configure.in仍通过旧名称调用 autoconf 宏的文件以使用当前的宏名称
ifnames	在configure.in为软件包编写文件时提供帮助；它会打印程序包在C预处理程序条件中使用的标识符[如果程序包已设置为具有一定的可移植性，则此程序可以帮助确定需要检查哪些 配置 。它还可以填补 自动扫描 configure.in生成的文件中的空白 。]

6.44。Automake-1.16.1

Automake软件包包含用于生成与Autoconf一起使用的Makefile的程序。

大概	编译
时间：	少于0.1 SBU（经测试约为8.7 SBU）
所需磁盘空间：	107 MB

6.44.1。安装Automake

准备Automake进行编译：

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.1
```

编译软件包：

```
make
```

由于各个测试的内部延迟，使用-j4 make选项可以加快测试速度，即使在只有一个处理器的系统上也是如此。要测试结果，请发出：

```
make -j4 check
```

已知一种测试在LFS环境中失败：subobj.sh。

安装软件包：

```
make install
```

6.44.2。Automake的内容

安装的程序： aclocal，aclocal-1.16（与aclocal硬链接），automake和automake-1.16（与automake硬链接）
安装目录： /usr/share/aclocal-1.16、/usr/share/automake-1.16和/usr/share/doc/automake-1.16.1

简短说明

本地	aclocal.m4 根据configure.in文件内容 生成文件
aclocal-1.16	与 aclocal 的硬链接
自动制作	一种Makefile.in从Makefile.am文件自动生成文件的 工具[要Makefile.in为包创建所有文件，请在顶层目录中运行该程序。通过扫描configure.in文件，它会自动找到每个合适的Makefile.am文件并生成相应的Makefile.in文件。]
自动制作1.16	的硬链接 automake的

6.45。Xz-5.2.4

Xz软件包包含用于压缩和解压缩文件的程序。它为lzma和更新的xz压缩格式提供功能。与传统的gzip或 bzip2命令相比，使用xz压缩文本文件产生的压缩百分比更高 。

时间： 预计编制 0.2 SBU
所需磁盘空间： 16 MB

6.45.1。安装Xz

准备Xz进行以下编译：

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.4
```

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包，并确保所有基本文件都在正确的目录中：

```
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的内容

已安装程序： 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

已安装的库： liblzma.so

安装目录： / usr / include / lzma和/usr/share/doc/xz-5.2.4

简短说明

猫咪	解压缩到标准输出
lzcmp	在LZMA压缩文件上 运行 cmp
伊兹迪夫	在LZMA压缩文件上 运行 diff
泽格列普	在LZMA压缩文件上 运行 egrep
lzfgrep	在LZMA压缩文件上 运行 fgrep
lzgrep	在LZMA压缩文件上 运行 grep
无zz	在LZMA压缩文件上 运行 较少
伊兹玛	使用LZMA格式压缩或解压缩文件
兹马德茨	小型，快速的LZMA压缩文件解码器
资讯网	显示存储在LZMA压缩文件头中的信息
伊兹莫尔	在LZMA压缩文件上 运行 更多
Unzzma	使用LZMA格式解压缩文件
x	使用XZ格式解压缩文件
z	使用XZ格式压缩或解压缩文件

cat猫	解压缩到标准输出
xzcmp	在XZ压缩文件上 运行 cmp
xzdec	小型，快速的XZ压缩文件解码器
xzdiff	在XZ压缩文件上 运行 diff
xzegrep	在XZ压缩文件上 运行 egrep
xzfgrep	在XZ压缩文件上 运行 fgrep
z	在XZ压缩文件上 运行 grep
xzless	在XZ压缩文件上 运行 较少
xzmore	在XZ压缩文件上 运行 更多
liblzma	该库使用Lempel-Ziv-Markov链算法实现无损块分类数据压缩

6.46。Kmod-26

Kmod软件包包含用于加载内核模块的库和实用程序

	预计编制
时间：	0.1 SBU
所需磁盘空间：	13 MB

6.46.1。安装Kmod

准备Kmod进行编译：

```
./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可以处理压缩的内核模块。

--with-rootlibdir=/lib
此选项可确保将与库相关的不同文件放置在正确的目录中。

编译软件包：

```
make
```

该软件包不附带可以在LFS chroot环境中运行的测试套件。至少需要git程序，并且一些测试不会在git存储库之外运行。

安装软件包，并创建符号链接以与Module-Init-Tools（以前处理Linux内核模块的软件包）兼容：

```
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的内容

已安装程序： 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）

已安装的库： libkmod.so

简短说明

德普莫德	根据在现有模块集中找到的符号创建依赖文件； modprobe 使用此依赖文件自动加载所需的模块
插入	在正在运行的内核中安装可加载模块
公里	加载和卸载内核模块
lsmod	列出当前加载的模块
modinfo	检查与内核模块关联的目标文件，并显示其可以收集的所有信息
Modprobe	使用由 depmod 创建的依赖文件自动加载相关模块
rmmod	从正在运行的内核中卸载模块
libkmod	其他程序使用此库来加载和卸载内核模块

6.47。Gettext-0.20.1

Gettext程序包包含用于国际化和本地化的实用程序。这些允许程序使用NLS（本机语言支持）进行编译，从而使它们能够以用户的母语输出消息。

	预计编制
时间：	2.9 SBU
所需磁盘空间：	249 MB

6.47.1。Gettext的安装

准备要编译的Gettext：

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.20.1
```

编译软件包：

```
make
```

要测试结果（这需要很长时间，大约需要3个SBU），请发出：

```
make check
```

安装软件包：

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

6.47.2。Gettext的内容

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和xgettext
已安装的库：	libasprintf.so，libgettextlib.so，libgettextpo.so，libgettextsrc.so和preloadable_libintl.so
安装目录：	/usr/lib/gettext、/usr/share/doc/gettext-0.20.1、/usr/share/gettext和/usr/share/gettext-0.19.8

简短说明

自动点	将标准Gettext基础结构文件复制到源包中
环境	用shell格式的字符串替换环境变量
文字	通过在消息目录中查找翻译，将自然语言消息翻译为用户的语言
gettext.sh	主要用作gettext的shell函数库
gettextize	将所有标准Gettext文件复制到包的给定顶级目录中，以开始对其进行国际化

msgattrib	根据属性过滤翻译目录中的消息并处理这些属性
消息猫	连接并合并给定的.po文件
msgcmp	比较两个.po 文件以检查两个文件是否包含相同的msgid字符串集
msgcomm	查找给定.po文件 共有的消息
msgconv	将翻译目录转换为其他字符编码
msgen	创建英文翻译目录
msgexec	将命令应用于翻译目录的所有翻译
msgfilter	将过滤器应用于翻译目录的所有翻译
msgfmt	从翻译目录生成二进制消息目录
msggrep	提取翻译目录中与给定模式匹配或属于某些给定源文件的所有消息
msginit	创建一个新.po 文件，使用来自用户环境的值初始化元信息
msgmerge	将两个原始翻译合并到一个文件中
msgunfmt	将二进制消息目录反编译为原始翻译文本
msguniq	统一翻译目录中的重复翻译
ngettext	显示文字消息的母语翻译，其语法形式取决于数字
recode-sr-latin	将塞尔维亚文字从西里尔文字重新编码为拉丁文字
xgettext	从给定的源文件中提取可翻译的消息行，以制作第一个翻译模板
libasprintf	定义 *autosprintf* 类，该类使C格式的输出例程可在C ++程序中使用，与 <*string*> 字符串和 <*iostream*> 流一起使用
libgettextlib	一个私有库，其中包含各种Gettext程序使用的通用例程；这些不适合一般使用
libgettextpo	用于编写处理.po文件的专用程序 ；当Gettext附带的标准应用程序（例如 msgcomm ， msgcmp ， msgattrib 和 msgen ） 不足时 ， 可以使用此库
libgettextsrc	一个私有库，其中包含各种Gettext程序使用的通用例程；这些不适合一般使用
preloadable_libintl	打算由LD_PRELOAD使用的库，它有助于libintl记录未翻译的消息

6.48。来自Elfutils-0.177的Libelf

Libelf是用于处理ELF（可执行和可链接格式）文件的库。

	预计编制
时间：	1.1 SBU
所需磁盘空间：	95 MB

6.48.1。Libelf的安装

Libelf是elfutils-0.177软件包的一部分。使用elfutils-0.177.tar.bz2作为源压缩包。

准备Libelf进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

仅安装Libelf：

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
```

6.48.2。Libelf的内容

已安装的库： libelf.so
安装目录： / usr / include / elfutils

6.49。Libffi-3.2.1

Libffi库为各种调用约定提供了一个可移植的高级编程接口。这允许程序员在运行时调用由调用接口描述指定的任何函数。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	7.6 MB

6.49.1。Libffi的安装

注意

与GMP相似，libffi会针对使用中的处理器进行优化。如果要为另一个系统构建，请导出CFLAGS和CXXFLAGS以为您的体系结构指定通用构建。如果不这样做，所有链接到libffi的应用程序将触发非法操作错误。

修改Makefile，以将标头安装到标准 /usr/include目录中，而不是/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
```

准备libffi进行编译：

```
./configure --prefix=/usr --disable-static --with-gcc-arch=native
```

configure选项的含义：

--with-gcc-arch=native
确保gcc针对当前系统进行了优化。如果未指定，则猜测系统，并且对于某些系统，生成的代码可能不正确。如果将生成的代码从本机系统复制到功能较弱的系统，请使用功能较弱的系统作为参数。有关其他系统类型的详细信息，请参见 **gcc手册中的x86选项**。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.49.2。利比菲的内容

已安装的库： libffi.so

简短说明

libffi 包含libffi API函数。

6.50。OpenSSL-1.1.1c

OpenSSL软件包包含与加密有关的管理工具和库。这些对于为其他软件包提供加密功能很有用，例如OpenSSH，电子邮件应用程序和Web浏览器（用于访问HTTPS站点）。

	预计编制
时间：	2.3 SBU
所需磁盘空间：	147 MB

6.50.1。安装OpenSSL

首先，解决上游发现的问题：

```
sed -i '/\} data/s/ =.*$/;\n    memset(\&data, 0, sizeof(data));/' \
  crypto/rand/rand_lib.c
```

准备要编译的OpenSSL：

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

编译软件包：

```
make
```

要测试结果，请发出：

```
make test
```

安装软件包：

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

如果需要，请安装文档：

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-1.1.1c
cp -vfr doc/* /usr/share/doc/openssl-1.1.1c
```

6.50.2。OpenSSL的内容

安装的程序： c_rehash和openssl
已安装的库： libcrypto。{so，a}和libssl。{so，a}
安装目录： / etc / ssl，/ usr / include / openssl，/ usr / lib / engines和/usr/share/doc/openssl-1.1.1c

简短说明

c_rehash	是一个 Perl 脚本，用于扫描目录中的所有文件，并将符号链接添加到其哈希值。
的openssl	是一个命令行工具，用于从外壳程序使用 OpenSSL 的加密库的各种加密功能。它可以用于 man 1 openssl 中记录的各种功能。
libcrypto.so	实现了各种Internet标准中使用的多种加密算法。该库提供的服务由 SSL，TLS和S / MIME 的 OpenSSL 实现使用 ，并且也已用于实现 OpenSSH ， OpenPGP 和其他加密标准。
libssl.so	实现传输层安全性（TLS v1）协议。它提供了丰富的API，可以通过运行 man 3 ssl 找到相关文档。

6.51。Python-3.7.4

Python 3软件包包含Python开发环境。它对于面向对象的编程，编写脚本，对大型程序进行原型设计或开发整个应用程序很有用。

	预计编制
时间：	1.3 SBU
所需磁盘空间：	399 MB

6.51.1。安装Python 3

准备要编译的Python：

```
./configure --prefix=/usr       \
            --enable-shared     \
            --with-system-expat \
            --with-system-ffi   \
            --with-ensurepip=yes
```

配置选项的含义：

--with-system-expat
使用此开关可以链接到Expat的系统版本 。

--with-system-ffi
通过此开关，可以针对libffi的系统版本进行 链接。

--with-ensurepip=yes
此开关启用构建pip和setuptools打包程序。

编译软件包：

```
make
```

该测试套件需要TK和X Windows会话，并且必须在BLFS中重新安装Python 3后才能运行。

安装软件包：

```
make install
chmod -v 755 /usr/lib/libpython3.7m.so
chmod -v 755 /usr/lib/libpython3.so
ln -sfv pip3.7 /usr/bin/pip3
```

安装命令的含义：

chmod -v 755 /usr/lib/libpython3.{7m.,}so
修复库的权限要与其他库一致。

如果需要，请安装预格式化的文档：

```
install -v -dm755 /usr/share/doc/python-3.7.4/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.7.4/html \
    -xvf ../python-3.7.4-docs-html.tar.bz2
```

文档安装命令的含义：

--no-same-owner **和-没有相同的权限**
确保已安装的文件具有正确的所有权和权限。如果没有这些选项，则使用tar将使用 上游创建者的值安装软件包文件。

6.51.2。Python 3的内容

已安装程序： 2to3，idle3，pip3，pydoc3，python3，python3-config和pyvenv
已安装的库： libpython3.7m.so和libpython3.so

已安装目录： /usr/include/python3.7m、/usr/lib/python3和/usr/share/doc/python-3.7.4

简短说明

2对3	是一个 Python 程序，可读取 Python 2.x 源代码并应用一系列修复程序将其转换为有效的 Python 3.x 代码。
闲置3	是一个包装脚本，可打开一个支持 Python的 GUI编辑器。要运行此脚本，您必须在Python之前安装 Tk ，以便构建Tkinter Python模块。
点3	Python的软件包安装程序。您可以使用pip从Python Package Index和其他索引安装软件包。
pydoc3	是 Python 文档工具。
python3	是一种解释性，交互式，面向对象的编程语言。
pyvenv	在一个或多个目标目录中 创建虚拟 Python 环境。

6.52。忍者1.9.0

Ninja是一个注重速度的小型构建系统。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	69 MB

6.52.1。忍者的安装

运行时，忍者通常并行运行最大数量的进程。默认情况下，这是系统上的内核数加2。在某些情况下，这可能会使CPU过热或导致系统内存不足。如果从命令行运行，则传递-jN参数将限制并行进程的数量，但是某些程序包会嵌入ninja的执行，而不会传递-j参数。

使用 下面的*可选*过程，允许用户通过环境变量NINJAJOBS限制并行进程的数量。 例如，设置：

```
出口NINJAJOBS = 4
```

将忍者限制为四个并行进程。

如果需要，通过运行以下命令来添加使用环境变量NINJAJOBS的功能：

```
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
```

用以下方法构建忍者：

```
python3 configure.py --bootstrap
```

构建选项的含义：

--bootstrap
此参数强制忍者为当前系统重建自身。

要测试结果，请发出：

```
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

安装软件包：

```
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。忍者的内容

安装的程序： ninja

简短说明

忍者 是忍者构建系统。

6.53。介子-0.51.1

Meson是一个开源构建系统，它不仅要非常快，而且更重要的是要尽可能地方便用户使用。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	28 MB

6.53.1。安装介子

使用以下命令编译介子：

```
python3 setup.py build
```

该软件包没有测试套件。

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，python3 setup.py install将各种文件（例如手册页）安装到Python Eggs中。使用指定的根目录位置，setup.py将这些文件安装到标准层次结构中。然后，我们可以复制层次结构，以便文件位于标准位置。

6.53.2。介子的内容

安装的程序： meson
安装目录： /usr/lib/python3.7/site-packages/meson-0.51.1-py3.7.egg-info和/usr/lib/python3.7/site-packages/mesonbuild

简短说明

介子 高生产率的构建系统

6.54。Coreutils-8.31

Coreutils软件包包含用于显示和设置基本系统特征的实用程序。

	预计编制
时间：	2.5 SBU
所需磁盘空间：	202 MB

6.54.1。安装Coreutils

POSIX要求Coreutils的程序即使在多字节语言环境中也能正确识别字符边界。以下修补程序修复了此违规问题和其他与国际化有关的错误。

```
patch -Np1 -i ../coreutils-8.31-i18n-1.patch
```

> **注意**
>
> 过去，此修补程序中发现了许多错误。向Coreutils维护人员报告新错误时，请首先检查如果没有此补丁，它们是否可复制。

禁止在某些计算机上可以永远循环的测试：

```
sed -i '/test.lock/s/^/#/' gnulib-tests/gnulib.mk
```

现在准备Coreutils进行编译：

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

配置选项的含义：

自动重新配置
: 此命令更新生成的配置文件，该文件与最新版本的automake一致。

FORCE_UNSAFE_CONFIGURE=1
: 此环境变量允许将程序包构建为root用户。

--enable-no-install-program=kill,uptime
: 此开关的目的是防止Coreutils安装将在以后由其他软件包安装的二进制文件。

编译软件包：

```
make
```

如果未运行测试套件，请 跳至" 安装软件包 "。

现在，测试套件已准备就绪，可以运行。首先，运行要以用户身份运行的测试*root*：

```
make NON_ROOT_USERNAME=nobody check-root
```

我们将以用户身份运行其余测试 *nobody*。但是，某些测试要求用户是多个组的成员。为了避免跳过这些测试，我们将添加一个临时组，并使用户成为该组*nobody*的一部分：

```
echo "dummy:x:1000:nobody" >> /etc/group
```

修复一些权限，以便非root用户可以编译和运行测试：

```
chown -Rv nobody .
```

现在运行测试。确保su环境中的PATH 包含/ tools / bin。

```
su nobody -s /bin/bash \
         -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

已知测试程序test-getlogin在部分构建的系统环境（如chroot环境）中失败，但如果在本章末运行，则通过。还已知测试程序tty.sh失败。

删除临时组：

```
sed -i '/dummy/d' /etc/group
```

安装软件包：

```
make install
```

将程序移至FHS指定的位置：

```
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,stty,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
```

LFS-Bootscripts软件包中的某些脚本取决于 head， nice， sleep和 touch。由于 /usr在启动的早期和后期可能无法使用，因此这些二进制文件需要位于根分区上才能保持FHS兼容性：

```
mv -v /usr/bin/{head,nice,sleep,touch} /bin
```

6.54.2。Coreutils的内容

已安装程序： [，b2sum，base32，base64，basename，basenc，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，粘贴，pathchk，pinky，pr，printenv，printf，ptx，pwd，readlink，realpath，rm，rmdir，runcon，seq，sha1sum，sha224sum，sha256sum，sha384sum，sha512sum，shred，shuf，睡眠，排序，分割，状态，stdbuf，stty，总和，同步，tac，尾巴，三通，测试，超时，触摸，tr，true，截断，tsort，tty，uname，unexpand，uniq，unlink，users，vdir，wc，who，whoami和yes

已安装的库： libstdbuf.so（在/ usr / libexec / coreutils中）
安装目录： / usr / libexec / coreutils

简短说明

base32	根据base32规范（RFC 4648）编码和解码数据
base64	根据base64规范（RFC 4648）编码和解码数据
b2和	打印或检查BLAKE2（512位）校验和
基本名	从文件名中剥离所有路径和给定的后缀
Basenc	使用各种算法对数据进行编码或解码
猫	将文件连接到标准输出
chcon	更改文件和目录的安全上下文
chgrp	更改文件和目录的组所有权
chmod	将每个文件的权限更改为给定模式；该模式可以是要进行的更改的符号表示，也可以是表示新权限的八进制数字
wn	更改文件和目录的用户和/或组所有权
chroot	以指定目录作为/目录 运行命令
cks	打印循环冗余校验（CRC）校验和以及每个指定文件的字节数
通讯	比较两个排序文件，在三列中输出唯一的行和常见的行
cp	复制文件
csplit	将给定文件拆分为几个新文件，根据给定的模式或行号将其分离，并输出每个新文件的字节数
切	打印线段，根据给定的字段或位置选择零件
日期	以给定的格式显示当前时间，或设置系统日期
dd	使用给定的块大小和计数复制文件，同时可以选择对文件进行转换
df	报告所有已挂载文件系统上或仅在保存所选文件的文件系统上可用（和已使用）的磁盘空间量
目录	列出每个给定目录的内容（与 ls 命令相同）
dircolors	输出命令以设置LS_COLOR环境变量以更改 ls 使用的配色方案
目录名	从文件名中删除非目录后缀
杜	报告当前目录，每个给定目录（包括所有子目录）或每个给定文件使用的磁盘空间量
回声	显示给定的字符串
环保	在修改后的环境中运行命令
扩大	将制表符转换为空格

expr	计算表达式
因子	显示所有指定整数的质数
假	不执行任何操作，未成功；它总是以指示失败的状态码退出
fmt	重新格式化给定文件中的段落
折	在给定文件中换行
团体	报告用户的组成员身份
头	打印每个给定文件的前十行（或给定行数）
主机名	报告主机的数字标识符（以十六进制表示）
ID	报告当前用户或指定用户的有效用户ID，组ID和组成员身份
安装	在设置文件的权限模式以及其所有者和组的情况下复制文件
加入	连接来自两个单独文件的具有相同联接字段的行
链接	创建具有给定名称的文件硬链接
ln	在文件之间建立硬链接或软（符号）链接
登录名	报告当前用户的登录名
ls	列出每个给定目录的内容
md5sum	报告或检查Message Digest 5（MD5）校验和
麦克迪尔	用给定名称创建目录
麦克菲	用给定的名称创建先进先出（FIFO），这是UNIX术语中的“命名管道”
麦克诺德	用给定名称创建设备节点；设备节点是字符特殊文件，块特殊文件或FIFO
mktemp	以安全的方式创建临时文件；在脚本中使用
MV	移动或重命名文件或目录
不错	运行计划优先级已修改的程序
nl	编号给定文件中的行
Nohup	运行不受挂断影响的命令，其输出重定向到日志文件
nproc	打印流程可用的处理单元数
numfmt	将数字转换为人类可读的字符串
od	以八进制和其他格式转储文件
糊	合并给定文件，并排依次连接相应的行，并用制表符分隔
Pathchk	检查文件名是否有效或可移植
小指	是轻量级的手指客户端；它报告有关给定用户的一些信息

公关	对文件进行分页和分栏打印
打印环境	打印环境
打印	根据给定的格式打印给定的参数，就像C printf函数一样
ptx	根据给定文件的内容生成一个排列索引，每个关键字都在其上下文中
密码	报告当前工作目录的名称
阅读链接	报告给定符号链接的值
真实路径	打印已解析的路径
R M	删除文件或目录
rmdir	如果目录为空，则删除目录
Runcon	使用指定的安全上下文运行命令
序列	打印给定范围内并以给定增量的数字序列
sha1sum	打印或检查160位安全哈希算法1（SHA1）校验和
sha224sum	打印或检查224位安全哈希算法校验和
sha256sum	打印或检查256位安全哈希算法校验和
sha384sum	打印或检查384位安全哈希算法校验和
sha512sum	打印或检查512位安全哈希算法校验和
切碎	用复杂的模式重复覆盖给定的文件，从而使数据恢复变得困难
uf	随机排列文本行
睡觉	在给定的时间内暂停
分类	对给定文件中的行进行排序
分裂	按大小或行数将给定文件拆分为多个部分
统计	显示文件或文件系统状态
stdbuf	对标准流运行具有更改的缓冲操作的命令
姿势	设置或报告终端线路设置
和	打印每个给定文件的校验和和块计数
同步	刷新文件系统缓冲区；它将更改的块强制到磁盘并更新超级块
TAC	反向连接给定文件
尾巴	打印每个给定文件的最后十行（或给定行数）
三通	从标准输入读取，同时写入标准输出和给定文件
测试	比较值并检查文件类型

暂停	有时间限制地运行命令
触摸	更改文件时间戳，将给定文件的访问和修改时间设置为当前时间；不存在的文件的长度为零
TR	翻译，压缩和删除标准输入中的给定字符
真正	无所事事，成功；它总是以指示成功的状态码退出
截短	将文件缩小或扩展到指定大小
Tsort	执行拓扑排序；它根据给定文件中的部分排序写一个完全排序的列表
tty	报告连接到标准输入的终端的文件名
ame	报告系统信息
展开	将空格转换为制表符
优衣库	丢弃除连续相同行之一之外的所有行
取消连结	删除给定的文件
使用者	报告当前登录的用户的名称
虚拟目录	与 ls -l 相同
厕所	报告每个给定文件的行数，字数和字节数，以及给出多个文件时的总行数
谁	报告谁登录
我是谁	报告与当前有效用户ID关联的用户名
是	重复输出“ y ”或给定的字符串，直到被杀死
libstdbuf	stdbuf 使用的 库

6.55。检查0.12.0

Check是C的单元测试框架。

大概时间： 编译 0.1 SBU（测试时约为3.7 SBU）
所需磁盘空间： 12 MB

6.55.1。安装支票

准备检查以进行编译：

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

构建包：

```
make
```

编译完成。要运行Check测试套件，请发出以下命令：

```
make check
```

请注意，Check测试套件可能需要相对较长的时间（最多4 SBU）。

安装软件包并修复脚本：

```
make docdir=/usr/share/doc/check-0.12.0 install
sed -i '1 s/tools/usr/' /usr/bin/checkmk
```

6.55.2。支票内容

安装的程序： checkmk
已安装的库： libcheck。{a，so}

简短说明

Checkmk	Awk脚本，用于生成用于Check单元测试框架的C单元测试
libcheck.{a,so}	包含允许从测试程序中调用Check的函数

6.56。Diffutils-3.7

Diffutils软件包包含显示文件或目录之间差异的程序。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	36 MB

6.56.1。安装Diffutils

准备Diffutils进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.56.2。Diffutils的内容

安装的程序： cmp，diff，diff3和sdiff

简短说明

cmp	比较两个文件并报告它们是否不同或在哪些字节中不同
差异	比较两个文件或目录，并报告文件中哪些行不同
差异3	逐行比较三个文件
斯迪夫	合并两个文件并交互输出结果

6.57。高克5.0.1

Gawk软件包包含用于处理文本文件的程序。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	47 MB

6.57.1。安装Gawk

首先，确保未安装一些不需要的文件：

```
sed -i 's/extras//' Makefile.in
```

准备Gawk进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

如果需要，请安装文档：

```
mkdir -v /usr/share/doc/gawk-5.0.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.0.1
```

6.57.2。高克的内容

已安装的程序： awk（链接到gawk），gawk和awk-5.0.1

已安装的库： filefuncs.so，fnmatch.so，fork.so，inplace.so，intdiv.so，ordchr.so，readdir.so，readfile.so，revoutput.so，revtwoway.so，rwarray.so和时间。因此（全部在/ usr / lib / gawk中）

安装目录： / usr / lib / gawk，/ usr / libexec / awk，/ usr / share / awk和/usr/share/doc/gawk-5.0.1

简短说明

awk	指向 gawk 的链接
高克	用于处理文本文件的程序；它是 awk 的GNU实现
gawk-5.0.1	与 gawk 的硬链接

6.58。Findutils-4.6.0

Findutils程序包包含用于查找文件的程序。提供这些程序是为了在目录树中进行递归搜索并创建，维护和搜索数据库（通常比递归查找要快，但是如果最近未更新数据库则不可靠）。

	预计编制
时间：	0.7 SBU
所需磁盘空间：	52 MB

6.58.1。安装Findutils

首先，禁止测试在某些计算机上可能永远循环：

```
sed -i 's/test-lock..EXEEXT.//' tests/Makefile.in
```

接下来，进行glibc-2.28和更高版本所需的一些修复：

```
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
```

准备Findutils进行编译：

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

配置选项的含义：

--localstatedir
此选项将定位数据库的位置更改为/var/lib/locate，它符合FHS。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

LFS-Bootscripts包中的某些脚本取决于 find。由于 /usr在启动的早期阶段可能不可用，因此该程序必须位于根分区上。该updatedb的脚本还需要进行修改，以纠正明确的路径：

```
mv -v /usr/bin/find /bin
sed -i 's|find:=${BINDIR}|find:=/bin|' /usr/bin/updatedb
```

6.58.2。Findutils的内容

已安装程序： 查找，定位，updatedb和xargs
安装目录： / var / lib / locate

简短说明

找	在给定的目录树中搜索符合指定条件的文件
定位	搜索文件名数据库并报告包含给定字符串或匹配给定模式的名称
更新b	更新 定位 数据库；它扫描整个文件系统（包括当前已安装的其他文件系统，除非告知您不这样做），并将找到的每个文件名放入数据库中
xargs	可用于将给定命令应用于文件列表

6.59。格罗夫1.22.4

Groff软件包包含用于处理和格式化文本的程序。

	预计编制
时间：	0.5 SBU
所需磁盘空间：	95 MB

6.59.1。安装Groff

Groff希望环境变量PAGE包含默认的纸张尺寸。对于美国用户而言，*PAGE=letter*是合适的。在其他地方，*PAGE=A4* 可能更合适。虽然在编译过程中配置了默认纸张尺寸，但以后可以通过在文件中回显“ A4 ”或“ letter ”来覆盖默认纸张尺寸/etc/papersize。

准备Groff进行编译：

```
PAGE=<paper_size> ./configure --prefix=/usr
```

该软件包不支持并行构建。编译软件包：

```
make -j1
```

该软件包没有测试套件。

安装软件包：

```
make install
```

6.59.2。Groff的内容

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，hptty，hptbbibi，，mmroff，neqn，nroff，pdfmom，pdfroff，pfbtops，pic，pic2graph，post-grohtml，preconv，pre-grohtml，引用，roff2dvi，roff2html，roff2pdf，roff2ps，roff2text，roff2x，soelim，tbl，tfmtodit，tr

安装目录： / usr / lib / groff和/usr/share/doc/groff-1.22.4、/usr/share/groff

简短说明

addftinfo	读取troff字体文件并添加 groff 系统 使用的一些其他字体度量信息
恋物癖	创建用于 groff 和 grops 的字体文件
化学	Groff预处理器，用于生成化学结构图
eqn	将嵌入在troff输入文件中的方程式描述编译成 troff 可以理解的命令
eqn2graph	将troff EQN（等式）转换为裁剪的图像
gdiffmk	标记groff / nroff / troff文件之间的差异
胶体	将以lilypond语言编写的活页乐谱转换为groff语言
金珀	groff的预处理程序，允许将perl代码添加到groff文件中
拼音	groff的前处理程序，允许在groff文件中添加类似于欧洲的汉语拼音。
grap2graph	将抓图转换为裁剪的位图图像
n	一个 groff的 gremlin文件预处理器
格罗德维	产生TeX dvi格式的 groff 驱动程序
off	groff文档格式化系统的前端；通常，它将运行 troff 程序和适合所选设备的后处理器
格罗夫	在X和tty终端上显示groff文件和手册页
rog	读取其中的文件和猜测 groff的 选项 -e，-man，-me，-mm，-ms，-p，-s，和-t需要打印的文件，并报告 groff的 ，包括那些选项命令
Grolbp	是佳能CAPSL打印机（LBP-4和LBP-8系列激光打印机）的 groff 驱动程序
Grolj4	是 groff 的驱动程序，可产生适用于HP LaserJet 4打印机的PCL5格式的输出
gropdf	将GNU troff 的输出转换为PDF
摸索	将GNU troff 的输出转换为PostScript
肮脏的	将GNU troff 的输出转换为适合类似打字机的设备的形式
hpftodit	从带有HP标签的字体指标文件中 创建用于 groff -Tlj4 的字体文件
印地文	创建针对文献数据库的倒排索引，用于与使用的文件 是指 ， lookbib 和 lkbib
kb	在书目数据库中搜索包含指定键的参考，并报告找到的所有参考
Lookbib	打印有关标准错误的提示（除非标准输入不是终端），从标准输入中读取包含一组关键字的行，在指定文件的书目数据库中搜索包含那些关键字的引用，并打印在标准输出，并重复此过程直到输入结束

毫米波	一个简单的 groff 预处理器
neqn	格式化美国信息交换标准码（ASCII）输出的方程式
恩罗夫	使用 groff 模拟 nroff 命令的脚本
pdfmom	是围绕groff的包装程序，可帮助从使用mom宏格式化的文件中生成PDF文档。
pdf文件	使用groff创建pdf文档
pfbtops	将PostScript字体.pfb格式转换为ASCII
图片	将嵌入在troff或TeX输入文件中的图片的描述编译成TeX或 troff可以 理解的命令
pic2graph	将PIC图转换为裁剪的图像
grohtml之后	将GNU troff 的输出转换为HTML
预转换	将输入文件的编码转换为GNU troff可以 理解的 格式
grohtml前	将GNU troff 的输出转换为HTML
参考	拷贝一个文件的内容到标准输出，除了线之间 。*[* 和 *]* 被解释为引用，和之间的线 *.R1* 和 *.R2* 被解释为命令用于引用将如何被处理的
roff2dvi	将roff文件转换为DVI格式
roff2html	将roff文件转换为HTML格式
roff2pdf	将roff文件转换为PDF
roff2ps	将roff文件转换为ps文件
roff2text	将roff文件转换为文本文件
roff2x	将roff文件转换为其他格式
索林	读取文件，并用上述 *文件* 的内容替换 *.so文件 格式的行*
tbl	嵌入式troff输入文件中为命令表的编译描述由理解 的troff
tfmtodit	创建用于 groff -Tdvi 的字体文件
鳟鱼	与Unix troff 高度兼容；通常应使用 groff 命令调用该命令，该命令还将以适当的顺序和适当的选项运行预处理器和后处理器

6.60。GRUB-2.04

GRUB软件包包含GRand Unified Bootloader。

	预计编制
时间：	0.8 SBU
所需磁盘空间：	161 MB

6.60.1。安装GRUB

准备GRUB进行编译：

```
./configure --prefix=/usr          \
            --sbindir=/sbin        \
            --sysconfdir=/etc      \
            --disable-efiemu       \
            --disable-werror
```

新的配置选项的含义：

--disable-werror
这使构建可以完成，同时提供最新Flex版本引入的警告。

--disable-efiemu
此选项通过禁用功能和测试LFS不需要的程序来最大程度地减少构建的内容。

编译软件包：

```
make
```

该软件包没有测试套件。

安装软件包：

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

在第8.4节"使用GRUB设置引导过程"中将讨论使用GRUB使LFS系统可引导。

6.60.2。GRUB的内容

安装的程序： 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-mkrelpath，grub-mkrescue，grub-mkstandalone，grub-ofpathname，grub-probe，grub-reboot，grub-render-label，grub-script-check ，grub-set-default，grub-sparc64-setup和grub-syslinux2cfg

安装目录： /usr/lib/grub、/etc/grub.d、/usr/share/grub和/ boot / grub（首次运行grub-install时）

简短说明

grub-bios设置	是用于grub-install的帮助程序

grub编辑	编辑环境块的工具
grub文件	检查FILE是否为指定类型。
grub-fstest	调试文件系统驱动程序的工具
grub-glue-efi	处理ia32和amd64 EFI图像并根据Apple格式粘贴它们。
grub安装	在驱动器上安装GRUB
grub-kbdcomp	将xkb布局转换为GRUB可以识别的布局的脚本
rub虫	HFS或HFS +文件上的Mac风格的祝福
grub-menulst2cfg	将GRUB Legacy menu.lst转换grub.cfg为供GRUB 2使用
grub-mkconfig	生成一个grub配置文件
grub-mkimage	制作GRUB的可启动映像
grub-mklayout	生成GRUB键盘布局文件
grub-mknetdir	准备一个GRUB netboot目录
grub-mkpasswd-pbkdf2	生成加密的PBKDF2密码以在引导菜单中使用
grub-mkrelpath	使系统路径名相对于其根目录
抢救	使GRUB的可启动映像适合于软盘或CDROM / DVD
grub-mkstandalone	生成一个独立的图像
路径名	是一个帮助程序，可打印GRUB设备的路径
rub探针	探查给定路径或设备的设备信息
重启	设置GRUB的默认启动项，仅用于下一次启动
grub-render-label	为Apple Mac渲染Apple .disk_label
grub-script-check	检查GRUB配置脚本是否存在语法错误
grub-set-default	设置GRUB的默认启动项
grub-sparc64设置	是用于grub-setup的帮助程序
grub-syslinux2cfg	将syslinux配置文件转换为grub.cfg格式

6.61。少于551

Less程序包包含一个文本文件查看器。

	预计编制
时间：	少于0.1 SBU

所需磁盘空间： 4.1 MB

6.61.1。安装少

少准备编译：

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

配置选项的含义：

--sysconfdir=/etc
此选项告诉程序包创建的程序查找/etc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。少内容

安装的程序： less，lessecho和lesskey

简短说明

减	文件查看器或寻呼机；它显示给定文件的内容，让用户滚动，查找字符串并跳转到标记
莱斯科	需要扩展元字符，例如 *和 *?* ，在Unix系统上的文件名中
小键	用于为指定的键绑定 少

6.62。Gzip 1.10

Gzip软件包包含用于压缩和解压缩文件的程序。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	20 MB

6.62.1。安装Gzip

准备Gzip进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

已知有两个测试在LFS环境中失败：help-version和zmore。

安装软件包：

```
make install
```

移动需要在根文件系统上的程序：

```
mv -v /usr/bin/gzip /bin
```

6.62.2。Gzip的内容

安装的程序： gunzip，gzexe，gzip，解压缩（带有gunzip的硬链接），zcat，zcmp，zdiff，zegrep，zfgrep，zforce，zgrep，zless，zmore和znew

简短说明

拉链	解压缩压缩文件
gzexe	创建自解压的可执行文件
gzip	使用Lempel-Ziv（LZ77）编码压缩给定文件
解压缩	解压缩压缩文件
cat猫	将给定的压缩文件解压缩到标准输出
zcmp	在压缩文件上 运行 cmp
差异	在压缩文件上 运行 diff

Zegrep	在压缩文件上 运行 egrep
zfgrep	在压缩文件上 运行 fgrep
zforce	.gz 在所有给定的压缩文件上 强制扩展，以使 gzip 不会再次压缩它们；当文件名在文件传输过程中被截断时，这很有用
zgrep	在压缩文件上 运行 grep
Zless	在压缩文件上 运行 更少
兹莫尔	在压缩文件上 运行 更多
新	将文件从 压缩 格式重新压缩为 gzip 格式，.Z以 .gz

6.63。IPRoute2-5.2.0

IPRoute2程序包包含用于基本和高级基于IPV4的网络的程序。

	预计编制
时间：	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程序将无法生成，因为它依赖于Berkeley DB，而后者未安装在LFS中。但是，仍将安装arpd目录和手册页。通过运行以下命令来防止这种情况。如果需要arpd二进制文件，则可以在BLFS手册中的 **http://www.linuxfromscratch.org/blfs/view/9.0/server/databases.html#db上**找到有关编译Berkeley DB的说明。

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

还需要禁用构建两个需要 **http://www.linuxfromscratch.org/blfs/view/9.0/postlfs/iptables.html的**模块。

```
sed -i 's/.m_ipt.o//' tc/Makefile
```

编译软件包：

```
make
```

该软件包没有有效的测试套件。

安装软件包：

```
make DOCDIR=/usr/share/doc/iproute2-5.2.0 install
```

6.63.2。IPRoute2的内容

安装的程序： 网桥，ctstat（链接到lnstat），genl，ifcfg，ifstat，ip，lnstat，nstat，routef，routel，rtacct，rtmon，rtpr，rtstat（链接到lnstat），ss和tc

安装目录： / etc / iproute2，/ usr / lib / tc和/usr/share/doc/iproute2-5.2.0，

简短说明

桥	配置网桥
ctstat	连接状态实用程序
gen	通用Netlink实用程序前端
ifcfg	ip命令的 外壳程序脚本包装器[请注意，它需要**http://www.skbuff.net/iputils/上** 的iputils包中的arping和 rdisk程序 。
ifstat	显示接口统计信息，包括按接口发送和接收的数据包数量
ip	主要可执行文件。它具有几种不同的功能： ip链接*<device>* 允许用户查看设备状态并进行更改 ip addr 允许用户查看地址及其属性，添加新地址和删除旧地址 ip neighbor 允许用户查看邻居绑定及其属性，添加新的邻居条目，以及删除旧的条目 ip rule 允许用户查看路由策略并进行更改 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隧道及其属性，并进行更改 ip maddr 允许用户查看多播地址及其属性，并进行更改 ip mroute 允许用户设置，更改或删除多播路由 ip monitor 允许用户连续监视设备，地址和路由的状态
Instat	提供Linux网络统计信息；它是旧版rtstat程序 的通用且功能更全面的替代品
统计	显示网络统计信息
路由	ip route的 一个组成部分。这用于刷新路由表
路线	ip route的 一个组成部分。这是用于列出路由表的
rtacct	显示内容 /proc/net/rt_acct
rtmon	路线监控实用程序
rtpr	将ip -o的输出转换回可读形式
实时统计	路线状态实用程序
ss	类似于netstat命令；显示活动连接
tc	流量控制可执行文件；这是针对服务质量（QOS）和服务等级（COS）实施的

tc qdisc 允许用户设置排队规则
tc类 允许用户基于排队规则安排来设置类
tc估计器 允许用户估计流入网络的网络流量
tc过滤器 允许用户设置QOS / COS数据包过滤
tc策略 允许用户设置QOS / COS策略

6.64。Kbd-2.2.0

Kbd软件包包含键表文件，控制台字体和键盘实用程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	36 MB

6.64.1。安装Kbd

Backspace和Delete键的行为在Kbd程序包中的各个键映射中不一致。以下修补程序解决了i386键映射的此问题：

```
patch -Np1 -i ../kbd-2.2.0-backspace-1.patch
```

修补后，Backspace键生成代码为127的字符，Delete键生成众所周知的转义序列。

删除多余的resizecons程序（它需要已废止的svgalib提供视频模式文件-正常使用时，使用setfont调整控制台的大小）及其手册页。

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

准备Kbd进行编译：

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr --disable-vlock
```

配置选项的含义：

--disable-vlock
此选项可防止构建vlock实用程序，因为它需要PAM库，而该库在chroot环境中不可用。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

> **注意**
>
> 对于某些语言（例如白俄罗斯语），Kbd软件包没有提供有用的键盘映射，其中 常规的“ by ”键盘映射采用ISO-8859-5编码，并且通常使用CP1251键盘映射。使用这种语言的用户必须单独下载有效的键盘映射。

如果需要，请安装文档：

```
mkdir -v       /usr/share/doc/kbd-2.2.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.2.0
```

6.64.2。Kbd的含量

已安装程序： chvt，deallocvt，dumpkeys，fgconsole，getkeycodes，kbdinfo，kbd_mode，kbdrate，loadkeys，loadunimap，mapscrn，openvt，psfaddtable（链接到psfxtable），psfgettable（链接到psfxtable），psfstablex（链接到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-2.2.0和/ usr / share / unimaps

简短说明

chvt	更改前台虚拟终端
解除分配	解除分配未使用的虚拟终端
转储键	转储键盘翻译表
fgconsole	打印活动虚拟终端的号码
getkeycodes	打印内核扫描码到键码的映射表
知识库信息	获取有关控制台状态的信息
kbd_mode	报告或设置键盘模式
千比特率	设置键盘重复和延迟率
加载键	加载键盘翻译表

loadunimap	加载内核Unicode到字体的映射表
mapscrn	一个过时的程序，用于将用户定义的输出字符映射表加载到控制台驱动程序中；现在这是通过 setfont 完成的
开放式	在新的虚拟终端（VT）上启动程序
psfaddtable	将Unicode字符表添加到控制台字体
psfgettable	从控制台字体中提取嵌入式Unicode字符表
psfstriptable	从控制台字体中删除嵌入式Unicode字符表
psfxtable	处理控制台字体的Unicode字符表
setfont	在控制台上更改增强型图形适配器（EGA）和视频图形阵列（VGA）字体
setkeycodes	加载内核扫描码到键码的映射表条目；如果键盘上有异常按键，这很有用
定居者	设置键盘标志和发光二极管（LED）
设定元模式	定义键盘元键处理
setvtrgb	在所有虚拟终端中设置控制台颜色图
showconsolefont	显示当前的EGA / VGA控制台屏幕字体
显示键	报告键盘上按下的键的扫描码，键码和ASCII码
unicode_start	将键盘和控制台置于UNICODE模式[除非您的键盘映射文件采用ISO-8859-1编码，否则请不要使用此程序。对于其他编码，此实用程序会产生错误的结果。]
unicode_stop	从UNICODE模式还原键盘和控制台

6.65。脂质管道1.5.1

Libpipeline软件包包含一个库，用于以灵活方便的方式操纵子流程的管道。

	预计编制
时间：	0.1 SBU
所需磁盘空间：	9.0 MB

6.65.1。Libpipeline的安装

准备Libpipeline进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.65.2。脂质管道的内容

已安装的库： libpipeline.so

简短说明

libpipeline	该库用于在子流程之间安全地构建管道

6.66。制作4.2.1

Make软件包包含一个用于编译软件包的程序。

	预计编制
时间：	0.6 SBU
所需磁盘空间：	13 MB

6.66.1。安装品牌

再次，解决由glibc-2.27和更高版本引起的错误：

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

准备进行编译：

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

编译软件包：

```
make
```

测试套件需要知道支持Perl文件的位置。我们使用环境变量来完成此任务。要测试结果，请发出：

```
make PERL5LIB=$PWD/tests/ check
```

安装软件包：

```
make install
```

6.66.2。品牌内容

安装程序： make

简短说明

使 自动确定包装的哪些部分需要（重新）编译，然后发出相关命令

6.67。补丁2.7.6

补丁程序包包含一个程序，该程序用于通过应用 通常由diff程序创建的“ 补丁程序 ”文件来修改或创建文件。

预计编制
时间： 0.2 SBU
所需磁盘空间： 13 MB

6.67.1。安装补丁

准备补丁进行编译：

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

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.67.2。补丁内容

安装的程序： 补丁

简短说明

补丁 根据补丁文件修改文件[补丁文件通常是使用 diff 程序创建的差异列表。通过将这些差异应用于原始文件， 补丁程序 会创建补丁程序版本。

6.68。Man-DB-2.8.6.1

Man-DB软件包包含用于查找和查看手册页的程序。

	预计编制
时间：	0.4 SBU
所需磁盘空间：	38 MB

6.68.1。Man-DB的安装

准备Man-DB进行编译：

```
./configure --prefix=/usr                        \
            --docdir=/usr/share/doc/man-db-2.8.6.1 \
            --sysconfdir=/etc                    \
            --disable-setuid                     \
            --enable-cache-owner=bin             \
            --with-browser=/usr/bin/lynx         \
            --with-vgrind=/usr/bin/vgrind        \
            --with-grap=/usr/bin/grap            \
            --with-systemdtmpfilesdir=           \
            --with-systemdsystemunitdir=
```

配置选项的含义：

--disable-setuid

这将禁用将man程序setuid设置为user *man*。

--enable-cache-owner=bin

这使得系统范围的缓存文件归用户bin所有。

--with-...

这三个参数用于设置一些默认程序。lynx是基于文本的Web浏览器（有关安装说明，请参阅BLFS），vgrind 将程序源转换为Groff输入，而 grap对于在Groff文档中排版图形很有用。该 vgrind把 和虎视眈眈 查看手册页通常不需要程序。它们不是LFS或BLFS的一部分，但是您可以在完成LFS之后自行安装它们。

--with-systemd...

这些参数可防止安装不需要的systemd目录和文件。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

6.68.2。LFS中的非英语手册页

下表显示了Man-DB假定/usr/share/man/<ll>将在其下安装的手册页进行编码的字符集。除此之外，Man-DB可以正确确定该目录中安装的手册页是否为UTF-8编码。

表6.1。旧版8位手册页的预期字符编码

语言（代码）	编码方式	语言（代码）	编码方式
丹麦文（da）	ISO-8859-1	克罗地亚语（小时）	ISO-8859-2
德文（de）	ISO-8859-1	匈牙利文（hu）	ISO-8859-2
英文（en）	ISO-8859-1	日语（ja）	EUC-JP
西班牙语（es）	ISO-8859-1	韩语（ko）	EUC-KR
爱沙尼亚语（et）	ISO-8859-1	立陶宛语（lt）	ISO-8859-13
芬兰文（fi）	ISO-8859-1	拉脱维亚语（lv）	ISO-8859-13
法语（法语）	ISO-8859-1	马其顿语（mk）	ISO-8859-5
爱尔兰文（ga）	ISO-8859-1	波兰文（pl）	ISO-8859-2
加利西亚语（gl）	ISO-8859-1	罗马尼亚语（ro）	ISO-8859-2

语言（代码）	编码方式	语言（代码）	编码方式
印尼文（id）	ISO-8859-1	俄文（ru）	KOI8-R
冰岛文（is）	ISO-8859-1	斯洛伐克文（sk）	ISO-8859-2
义大利文（it）	ISO-8859-1	斯洛文尼亚文（sl）	ISO-8859-2
挪威博克马尔（nb）	ISO-8859-1	塞尔维亚拉丁语（sr @ latin）	ISO-8859-2
荷兰语（NL）	ISO-8859-1	塞尔维亚文（sr）	ISO-8859-5
挪威尼诺斯克（nn）	ISO-8859-1	土耳其文（tr）	ISO-8859-9
挪威文（否）	ISO-8859-1	乌克兰文（英国）	KOI8-U
葡萄牙语（pt）	ISO-8859-1	越南文（vi）	TCVN5712-1
瑞典语（SV）	ISO-8859-1	简体中文（zh_CN）	GBK
白俄罗斯语（be）	CP1251	简体中文，新加坡（zh_SG）	GBK
保加利亚（bg）	CP1251	繁体中文，香港（zh_HK）	BIG5HKSCS
捷克文（cs）	ISO-8859-2	繁体中文（zh_TW）	大5
希腊文（el）	ISO-8859-7		

注意

不支持列表中未包含语言的手册页。

6.68.3。Man-DB的内容

安装的程序： accessdb，apropos（链接到whatis），catman，lexgrog，man，mandb，manpath和whatis
已安装的库： libman.so和libmandb.so（均在/ usr / lib / man-db中）
安装目录： / usr / lib / man-db，/ usr / libexec / man-db和/usr/share/doc/man-db-2.8.6.1

简短说明

访问数据库	以易于阅读的形式 转储 whatis 数据库内容
适当的	搜索 whatis 数据库并显示包含给定字符串的系统命令的简短描述
猫人	创建或更新预格式化的手册页
莱格罗格	显示有关给定手册页的单行摘要信息
男子	格式化并显示所需的手册页
mandb	创建或更新 whatis 数据库
人行道	根据man.conf中的设置和用户环境显示$ MANPATH的内容或（如果未设置$ MANPATH）显示合适的搜索路径的内容
什么是	搜索 whatis 数据库并显示包含给定关键字作为单独单词的系统命令的简短描述
libman	包含对 人的 运行时支持
libmandb	包含对 人的 运行时支持

6.69。焦油1.32

Tar软件包包含一个归档程序。

	预计编制
时间：	2.2 SBU
所需磁盘空间：	45 MB

6.69.1。安装焦油

准备Tar以进行编译：

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr \
            --bindir=/bin
```

配置选项的含义：

FORCE_UNSAFE_CONFIGURE=1

这将强制测试mknod以root用户身份运行。通常，以root用户身份运行此测试很危险，但是由于它是在仅部分构建的系统上运行，因此可以覆盖它。

编译软件包：

```
make
```

要测试结果（大约3个SBU），请发出：

```
make check
```

安装软件包：

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.32
```

6.69.2。焦油含量

安装的程序： tar
安装目录： /usr/share/doc/tar-1.32

简短说明

柏油 从档案创建，提取文件并列出档案的内容，也称为tarball

6.70。Texinfo-6.6

Texinfo软件包包含用于读取，写入和转换信息页面的程序。

预计编制
时间： 0.8 SBU
所需磁盘空间： 122 MB

6.70.1。安装Texinfo

准备Texinfo进行编译：

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

配置选项的含义：

--disable-static
在这种情况下，顶级configure脚本将抱怨这是无法识别的选项，但是XSParagraph的configure脚本可以识别它，并使用它禁用将静态对象安装XSParagraph.a到/usr/lib/texinfo。

编译软件包：

```
make
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

（可选）安装属于TeX安装的组件：

```
make TEXMF=/usr/share/texmf install-tex
```

make参数的含义：

TEXMF=/usr/share/texmf
在TEXMF生成文件变量保存TeX的树的根的位置，如果，例如，TeX的将在后面安装。

信息文档系统使用纯文本文件保存其菜单项列表。该文件位于/usr/share/info/dir。不幸的是，由于各种软件包的Makefile偶尔出现问题，它有时可能与系统上安装的信息页面不同步。如果/usr/share/info/dir需要重新创建文件，则以下可选命令将完成任务：

```
pushd /usr/share/info
rm -v dir
for f in *
  do install-info $f dir 2>/dev/null
done
popd
```

6.70.2。Texinfo的内容

安装的程序： info，install-info，makeinfo（指向texi2any的链接），pdftexi2dvi，pod2texi，texi2any，texi2dvi，texi2pdf和texindex
已安装的库： MiscXS.so，Parsetexi.so和XSParagraph.so（均在/ usr / lib / texinfo中）
安装目录： / usr / share / texinfo和/ usr / lib / texinfo

简短说明

信息	用于读取与手册页相似的信息页，但通常比仅解释所有可用的命令行选项更深入[例如，比较 man bison 和 info bison 。]
安装信息	用于安装信息页面；它更新 信息 索引文件中的 条目
makeinfo	将给定的Texinfo源文档翻译成信息页，纯文本或HTML

pdftexi2dvi	用于将给定的Texinfo文档格式化为可移植文档格式（PDF）文件
pod2texi	将Pod转换为Texinfo格式
texi2any	将Texinfo源文档翻译成其他各种格式
texi2dvi	用于将给定的Texinfo文档格式化为与设备无关的文件，可以打印该文件
texi2pdf	用于将给定的Texinfo文档格式化为可移植文档格式（PDF）文件
特克斯指数	用于对Texinfo索引文件进行排序

6.71。Vim-8.1.1846

Vim软件包包含一个功能强大的文本编辑器。

	预计编制
时间：	2.2 SBU
所需磁盘空间：	190 MB

Vim的替代品

如果您喜欢其他编辑器（例如Emacs，Joe或Nano），请参考 **http://www.linuxfromscratch.org/blfs/view/9.0/postlfs/editors.html** 以获得建议的安装说明。

6.71.1。安装Vim

首先，将vimrc配置文件的默认位置更改为/etc：

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

准备Vim进行编译：

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

编译软件包：

```
make
```

要准备测试，请确保*nobody*用户可以写入源代码树：

```
chown -Rv nobody .
```

现在以*nobody*用户身份运行测试：

```
su nobody -s /bin/bash -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

测试套件将大量二进制数据输出到屏幕。这可能会导致当前终端的设置出现问题。如上所示，可以通过将输出重定向到日志文件来避免该问题。成功的测试将在完成时在日志文件中显示单词" ALL DONE"。

安装软件包：

```
make install
```

许多用户习惯于使用vi代替vim。要在用户习惯性地输入vi时允许执行 vim，请使用提供的语言为二进制文件和手册页创建符号链接：

```
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的文档安装在中/usr/share/vim。以下符号链接允许通过来访问文档/usr/share/doc/vim-8.1.1846，使其与其他软件包的文档位置一致：

```
ln -sv ../vim/vim81/doc /usr/share/doc/vim-8.1.1846
```

如果要在LFS系统上安装X窗口系统，则在安装X之后可能需要重新编译Vim。Vim带有编辑器的GUI版本，需要安装X和一些其他库。有关此过程的更多信息，请参考BLFS手册中的Vim文档和Vim安装页面，网址为**http://www.linuxfromscratch.org/blfs/view/9.0/postlfs/vim.html**。

6.71.2。配置Vim

默认情况下，vim 在vi不兼容模式下运行。对于过去使用其他编辑器的用户而言，这可能是新的。该" nocompatible "设置包括高亮显示正在使用一种新的行为的事实。它还提醒那些将更改为" 兼容 "模式的用户，它应该是配置文件中的第一个设置。这是必需的，因为它会更改其他设置，并且必须在此设置之后进行覆盖。通过运行以下命令创建默认的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
```

该*set nocompatible*设置使vim的行为比与vi兼容的行为更为有用（默认）。删除“ no ”以保留旧的vi行为。该*set backspace=2*设置允许在换行符，自动缩进和插入开始时退格。该*syntax on*参数启用vim的语法突出显示。*set mouse=*在chroot或通过远程连接进行操作时，该设置可使用鼠标正确粘贴文本。最后，带有设置的*if*语句*set background=dark*可以更正vim关于某些终端仿真器的背景色的猜测。这为突出显示了在这些程序的黑色背景上使用的更好的配色方案。

可通过运行以下命令来获取其他可用选项的文档：

```
vim -c ':options'
```

注意

默认情况下，Vim仅安装英语的拼写文件。要安装适用于您首选语言的拼写文件，请 从**ftp://ftp.vim.org/pub/vim/runtime/spell/下载** 适用于您的语言和字符编码*.spl 的*.sug文件， 然后将它们保存到。 /usr/share/vim/vim81/spell/

要使用这些拼写文件，/etc/vimrc需要进行一些配置 ，例如：

```
set spelllang=en,ru
set spell
```

有关更多信息，请参见上面的URL上的相应自述文件。

6.71.3。Vim的内容

安装的程序： 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

安装目录： / usr / share / vim

简短说明

前 在ex模式下 启动 vim

查看	是 视图 的受限版本; 无法启动任何Shell命令，并且无法挂起 视图
im	是 vim 的限制版本; 无法启动任何shell命令，并且无法挂起 vim
六	链接到 vim
视图	以只读模式 启动 vim
vim	是编辑
维姆迪夫	使用 vim 编辑文件的两个或三个版本 并显示差异
维姆托	讲解 vim 的基本键和命令
xxd	创建给定文件的十六进制转储；它也可以相反，因此可以用于二进制修补

6.72。Procps-ng-3.3.15

Procps-ng软件包包含用于监视过程的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	17 MB

6.72.1。安装程序

准备procps-ng进行编译：

```
./configure --prefix=/usr                            \
            --exec-prefix=                           \
            --libdir=/usr/lib                        \
            --docdir=/usr/share/doc/procps-ng-3.3.15 \
            --disable-static                         \
            --disable-kill
```

配置选项的含义：

--disable-kill
此开关禁用构建将由Util-linux软件包安装的kill命令。

编译软件包：

```
make
```

该测试套件需要对LFS进行一些自定义修改。删除脚本不使用tty设备时失败的测试，并修复另外两个。要运行测试套件，请运行以下命令：

```
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，将基本库移到/usr未安装的位置。

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

6.72.2。处理内容

已安装程序： 免费，pgrep，pidof，pkill，pmap，ps，pwdx，slabtop，sysctl，tload，top，正常运行时间，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

简短说明

自由	报告系统中的可用和已用内存量（物理内存和交换内存）
pgrep	根据名称和其他属性查找流程
皮多夫	报告给定程序的PID
杀人	根据进程的名称和其他属性发出信号
pmap	报告给定进程的内存映射
ps	列出当前正在运行的进程
密码	报告进程的当前工作目录
平板电脑	实时显示详细的内核Slab缓存信息
系统控制	在运行时修改内核参数
负荷	打印当前系统平均负载的图形
最佳	显示最消耗CPU的进程的列表；它可以实时查看处理器活动
正常运行时间	报告系统已运行多长时间，已登录多少用户以及平均系统负载
虚拟机	报告虚拟内存统计信息，提供有关进程，内存，页面调度，块输入/输出（IO），陷阱和CPU活动的信息

w	显示当前登录的用户，登录时间和地点
看	重复运行给定命令，显示其输出的第一个屏幕；这使用户可以观察输出随时间的变化
libprocps	包含此程序包中大多数程序使用的功能

6.73。Util-linux-2.34

Util-linux软件包包含其他实用程序。其中包括用于处理文件系统，控制台，分区和消息的实用程序。

	预计编制
时间：	1.2 SBU
所需磁盘空间：	283 MB

6.73.1。FHS合规说明

FHS建议使用/var/lib/hwclock目录而不是通常的/etc目录作为adjtime文件的位置。首先创建一个目录以启用hwclock 程序的存储 ：

```
mkdir -pv /var/lib/hwclock
```

6.73.2。安装Util-linux

准备要编译的Util-linux：

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --docdir=/usr/share/doc/util-linux-2.34 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            --without-systemd    \
            --without-systemdsystemunitdir
```

--disable和--without选项可防止发出警告，提示构建不需要LFS软件包或与其他软件包安装的程序不一致的组件。

编译软件包：

```
make
```

如果需要，请以非root用户身份运行测试套件：

警告

以root用户身份运行测试套件可能对您的系统有害。要运行它，内核的CONFIG_SCSI_DEBUG选项必须在当前运行的系统中可用，并且必须作为模块构建。将其构建到内核中将阻止启动。为了完全覆盖，必须安装其他BLFS软件包。如果需要，可以在重新启动到完整的LFS系统并运行后运行此测试：

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

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

安装软件包：

```
make install
```

6.73.3。Util-linux的内容

已安装程序： addpart，agetty，blkdiscard，blkid，blkzone，blockdev，cal，cfdisk，chcpu，chmem，choom，chrt，col，colcrt，colrm，列，ctrlaltdel，delpart，dmesg，弹出，fallocate，fdformat，fdisk，fincore，findfs， findmnt，flock，fsck，fsck.cramfs，fsck.minix，fsfreeze，fstrim，getopt，hexdump，hwclock，i386，ionice，ipcmk，ipcrm，ipcs，isosize，kill，last，lastb（链接到最后），ldattach，linux32 ，linux64，logger，look，lostup，lsblk，lscpu，lsipc，lslocks，lslogins，lsmem，lsns，mcookie，mesg，mkfs，mkfs.bfs，mkfs.cramfs，mkfs.minix，mkswap，更多，mount，mountpoint， ，nsenter，partx，pivot_root，prlimit，raw，readprofile，重命名，renice，resizepart，rev，rfkill，rtcwake，脚本，scriptreplay，setarch，setsid，setterm，sfdisk，sulogin，swaplabel，swapoff（链接到swapon），swapon，switch_root，taskset，ul，umount，uame26，uname26，unshare，utmpdump，uuidd ，uuidgen，uuidparse，wall，wdctl，whereis，wipefs，x86_64和zramctl

已安装的库： libblkid.so，libfdisk.so，libmount.so，libsmartcols.so和libuuid.so

安装目录： /usr/include/blkid、/usr/include/libfdisk、/usr/include/libmount、/usr/include/libsmartcols、/usr/include/uuid、/usr/share/doc/util-linux-2.34和/ var / lib / hwclock

简短说明

加法	通知Linux内核新分区
Agetty	打开tty端口，提示输入登录名，然后调用 登录 程序
blkdiscard	丢弃设备上的扇区

笨蛋	命令行实用程序，用于查找和打印块设备属性
blkzone	在给定的块设备上运行区域命令
块开发	允许用户从命令行调用块设备ioctl
卡	显示一个简单的日历
cfdisk	操作给定设备的分区表
chcpu	修改CPU的状态
chmem	配置内存
合唱	显示并调整OOM杀手得分
chrt	操作流程的实时属性
关口	过滤掉反向换行
科尔特	过滤 nroff 输出以用于缺少某些功能的终端，例如超行程和半行
科尔姆	过滤出给定的列
柱	将给定文件格式化为多列
ctrlaltdel	将Ctrl + Alt + Del组合键的功能设置为硬复位或软复位
拆装	要求Linux内核删除分区
dmesg	转储内核启动消息
喷射	弹出可移动媒体
堕落	将空间预分配给文件
fdformat	低级格式化软盘
磁盘	操作给定设备的分区表
芬科	计算核心中文件内容的页面
findfs	通过标签或通用唯一标识符（UUID）查找文件系统
寻找	是libmount库的命令行界面，可与mountinfo，fstab和mtab文件一起使用
群	获取文件锁，然后在保持锁的状态下执行命令
fsck	用于检查和可选地修复文件系统
fsck.cramfs	在给定设备上的Cramfs文件系统上执行一致性检查
fsck.minix	在给定设备上的Minix文件系统上执行一致性检查
fsfreeze	是关于FIFREEZE / FITHAW ioctl内核驱动程序操作的非常简单的包装器
fstrim	丢弃已挂载文件系统上未使用的块
getopt	解析给定命令行中的选项

十六进制转储	以十六进制或其他给定格式转储给定文件
时钟	读取或设置系统的硬件时钟，也称为实时时钟（RTC）或基本输入输出系统（BIOS）时钟
i386	指向setarch的符号链接
离子离子	获取或设置程序的io调度类和优先级
ipcmk	创建各种IPC资源
ipcrm	删除给定的进程间通信（IPC）资源
ipcs	提供IPC状态信息
等尺寸	报告iso9660文件系统的大小
杀	向过程发送信号
持续	显示哪些用户最后一次登录（和注销），并在/var/log/wtmp文件中进行搜索；它还显示了系统启动，关闭和运行级别更改
拉斯特	显示登录失败的尝试 /var/log/btmp
ldattach	将线路规则附加到串行线路
linux32	指向setarch的符号链接
linux64	指向setarch的符号链接
记录器	将给定的消息输入系统日志
看	显示以给定字符串开头的行
输了	设置和控制回路设备
lsblk	以树状格式列出有关所有或所选块设备的信息
lscpu	打印CPU架构信息
电脑	在系统中当前使用的IPC设施上打印信息
锁	列出本地系统锁
登录	列出有关用户，组和系统帐户的信息
记忆	列出可用内存的范围及其在线状态
lsns	列出名称空间
饼干	为 xauth 生成魔术cookie（128位随机十六进制数字）
消息	控制其他用户是否可以将消息发送到当前用户的终端
mkfs	在设备（通常是硬盘分区）上构建文件系统
mkfs.bfs	创建一个Santa Cruz Operations（SCO）bfs文件系统
mkfs.cramfs	创建一个cramfs文件系统
mkfs.minix	创建一个Minix文件系统

mkswap	初始化给定的设备或文件以用作交换区域
更多	一次在一个屏幕上翻阅文本的过滤器
安装	将给定设备上的文件系统附加到文件系统树中的指定目录
挂载点	检查目录是否为挂载点
纳美	显示给定路径名中的符号链接
恩森特	使用其他进程的名称空间运行程序
零件	告诉内核磁盘上分区的存在和编号
ivot_root	使给定的文件系统成为当前进程的新根文件系统
限制	获取并设置流程的资源限制
生的	将Linux原始字符设备绑定到块设备
阅读资料	读取内核分析信息
改名	重命名给定的文件，用另一个替换给定的字符串
伦尼斯	更改正在运行的进程的优先级
调整大小	要求Linux内核调整分区大小
转速	反转给定文件的行
rkfill	启用和禁用无线设备的工具
rtcwake	用于进入系统睡眠状态直到指定的唤醒时间
脚本	制作终端会话的打字稿
脚本重播	使用时间信息播放打字稿
Setarch	在新的程序环境中更改报告的体系结构并设置个性标志
setid	在新会话中运行给定程序
设定项	设置终端属性
磁盘	磁盘分区表操纵器
素	允许*root*登录；通常在系统进入单用户模式时 由 init 调用
交换标签	允许更改交换区域UUID和标签
掉期	禁用设备和文件以进行分页和交换
交换	启用设备和文件以进行分页和交换，并列出当前正在使用的设备和文件
switch_root	切换到另一个文件系统作为挂载树的根
尾巴	跟踪日志文件的增长；显示日志文件的最后10行，然后在创建时继续在日志文件中显示所有新条目
任务集	检索或设置进程的CPU关联性

ul	用于将下划线转换为转义序列的过滤器，用于指示所用终端的下划线
数量	断开文件系统与系统文件树的连接
uname26	指向setarch的符号链接
取消分享	运行带有父级未共享的某些名称空间的程序
utmpdump	以更用户友好的格式显示给定登录文件的内容
uid	UUID库使用的守护程序，以安全且有保证的唯一方式生成基于时间的UUID
乌伊德根	创建新的UUID。在过去和将来，在本地系统和其他系统上创建的所有UUID中，每个新的UUID都可以合理地认为是唯一的
uuidparse	解析唯一标识符的实用程序
壁	在所有当前登录用户的终端上显示文件或默认情况下的标准输入的内容
wdctl	显示硬件看门狗状态
哪里	报告给定命令的二进制，源和手册页的位置
纸巾	从设备擦除文件系统签名
x86_64	指向setarch的符号链接
Zramctl	设置和控制zram（压缩ram磁盘）设备的程序
libblkid	包含用于设备识别和令牌提取的例程
libfdisk	包含用于处理分区表的例程
libmount	包含用于块设备安装和卸载的例程
libsmartcols	包含用于以表格形式帮助屏幕输出的例程
libuuid	包含用于为可能在本地系统以外访问的对象生成唯一标识符的例程

6.74。E2fsprogs-1.45.3

E2fsprogs软件包包含用于处理*ext2*文件系统的实用程序 。它还支持*ext3*和 *ext4*日志文件系统。

	预计编制
时间：	3.1 SBU
所需磁盘空间：	108 MB

6.74.1。E2fsprogs的安装

E2fsprogs文档建议将该软件包构建在源树的子目录中：

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

准备E2fsprogs进行编译：

```
../configure --prefix=/usr           \
             --bindir=/bin           \
             --with-root-prefix=""   \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

环境变量和配置选项的含义：

--with-root-prefix="" **和** *--bindir=/bin*

某些程序（例如e2fsck程序）被认为是必不可少的程序。例如，当 /usr未安装时，这些程序仍然需要可用。他们属于在类似的目录/lib 和/sbin。如果未将此选项传递给E2fsprogs的configure，则程序将安装到该/usr 目录中。

--enable-elf-shlibs

这将创建此程序包中的某些程序使用的共享库。

*--disable-**

这将阻止E2fsprogs构建和安装*libuuid*and *libblkid*库， *uuidd*守护程序和 fsck 包装程序，因为Util-Linux将安装最新版本。

编译软件包：

```
make
```

要运行测试，请发出：

```
make check
```

E2fsprogs测试之一将尝试分配256 MB内存。如果您没有明显更多的RAM，请确保为测试启用足够的交换空间。有关创建和启用交换空间的详细信息，请参见 第2.5节"在分区上创建文件系统"和 第2.7节"安装新分区"。

安装二进制文件，文档和共享库：

```
make install
```

安装静态库和头文件：

```
make install-libs
```

使已安装的静态库可写，以便以后可以删除调试符号：

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

该软件包将安装gzip压缩.info文件，但不会更新系统范围的dir文件。解压缩该文件，然后dir使用以下命令更新系统文件：

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

如果需要，通过发出以下命令来创建并安装一些其他文档：

```
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
```

6.74.2。E2fsprogs的内容

已安装程序： badblocks，chatr，compile_et，debugfs，dumpe2fs，e2freefrag，e2fsck，e2image，e2label，e2mmpstatus，e2scrub，e2scrub_all，e2undo，e4crypt，e4defrag，filefrag，fsck.ext2，fsck.ext4，fsck.ext3，，mk_cmds，mke2fs，mkfs.ext2，mkfs.ext3，mkfs.ext4，mklost + found，resize2fs和tune2fs

已安装的库： libcom_err.so，libe2p.so，libext2fs.so和libss.so

安装目录： / usr / include / e2p，/ usr / include / et，/ usr / include / ext2fs，/ usr / include / ss，/ usr / lib / e2fsprogs，/ usr / share / et和/ usr / share / ss

简短说明

坏块	在设备（通常是磁盘分区）上搜索坏块
聊天室	更改*ext2*文件系统上文件的属性；它也改变*ext3*文件系统的版本日志*ext2*文件系统
compile_et	错误表编译器；它将错误代码名称和消息表转换为适合与com_err库 一起使用的C源文件
调试文件	文件系统调试器；它可用于检查和更改*ext2*文件系统 的状态
dumpe2fs	打印给定设备上存在的文件系统的超级块和块组信息
e2freefrag	报告可用空间碎片信息
e2fsck	用于检查和可选地修复 *ext2*文件系统和*ext3*文件系统
e2image	用于将关键*ext2*文件系统数据保存到文件中
e2label	显示或更改*ext2*给定设备上存在 的文件系统上的文件系统标签
e2mmpstatus	检查ext4文件系统的MMP状态

e2scrub	检查已安装的ext [234]文件系统的内容
e2scrub_all	检查所有已安装的ext [234]文件系统是否存在错误
e2undo	重放设备上找到的ext2 / ext3 / ext4文件系统的撤消日志undo_log [可用于通过e2fsprogs程序撤消失败的操作。]
e4crypt	Ext4文件系统加密实用程序
e4defrag	ext4文件系统的在线碎片整理程序
文件片段	报告特定文件碎片的严重程度
fsck.ext2	默认情况下检查*ext2*文件系统，并且是到 e2fsck 的硬链接
fsck.ext3	默认情况下检查*ext3*文件系统，并且是到 e2fsck 的硬链接
fsck.ext4	默认情况下检查*ext4*文件系统，并且是到 e2fsck 的硬链接
日志保存	将命令的输出保存在日志文件中
萨特	列出第二个扩展文件系统上文件的属性
mk_cmds	将命令名称和帮助消息表转换为适合与libss子系统库一起 使用的C源文件
mke2fs	在给定设备上 创建一个*ext2*或 *ext3*文件系统
mkfs.ext2	默认情况下创建*ext2*文件系统，并且是 mke2fs 的硬链接
mkfs.ext3	默认情况下创建*ext3*文件系统，并且是 mke2fs 的硬链接
mkfs.ext4	默认情况下创建*ext4*文件系统，并且是 mke2fs 的硬链接
mklost + found	用于lost+found在*ext2*文件系统上创建目录 ；它将磁盘块预分配到此目录以减轻 e2fsck 的任务
resize2fs	可用于放大或缩小*ext2*文件系统
tune2fs	调整的可调谐文件系统参数 *ext2*文件系统
libcom_err	常见错误显示例程
libe2p	由 dumpe2fs ， chattr 和 lsattr使用
libext2fs	包含使用户级程序能够操纵*ext2* 文件系统的 例程
libss	由 debugfs使用

6.75。Sysklogd-1.5.1

Sysklogd软件包包含用于记录系统消息的程序，例如发生异常情况时内核给出的消息。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	0.6 MB

6.75.1。Sysklogd的安装

首先，在klogd中修复在某些情况下导致分段错误的问题，并修复过时的程序构造：

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

编译软件包：

```
make
```

该软件包没有测试套件。

安装软件包：

```
make BINDIR=/sbin install
```

6.75.2。配置Sysklogd

/etc/syslog.conf 通过运行以下命令 创建一个新文件：

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

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

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

6.75.3。Sysklogd的内容

已安装程序： klogd和syslogd

简短说明

用于拦截和记录内核消息的系统守护程序

博客	
系统日志	记录系统程序提供的用于记录日志的消息[每条记录的消息至少包含一个日期戳和一个主机名，通常也包含程序的名称，但这取决于告知记录守护程序的方式。

6.76。Sysvinit-2.95

Sysvinit软件包包含用于控制系统的启动，运行和关闭的程序。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	1.3 MB

6.76.1。Sysvinit的安装

首先，应用一个修补程序，该修补程序删除其他软件包安装的几个程序，澄清一条消息，并修复编译器警告：

```
patch -Np1 -i ../sysvinit-2.95-consolidated-1.patch
```

编译软件包：

```
make
```

该软件包没有测试套件。

安装软件包：

```
make install
```

6.76.2。Sysvinit的内容

安装的程序： bootlogd，fstab解码，停止，初始化，killall5，关机（链接到停止），重新启动（链接到停止），运行级别，关机和telinit（链接到init）

简短说明

引导日志	将启动消息记录到日志文件中
fstab解码	使用fstab编码的参数运行命令
停止	通常使用该 选项调用 `shutdown -h`，除非已处于运行级别0，然后它告诉内核暂停系统；否则，它将通知内核停止系统。它在文件中 `/var/log/wtmp`指出系统已关闭

在里面	内核初始化硬件后将开始的第一个过程，该硬件接管引导过程并启动其配置文件中指定的所有过程
killall5	向所有进程发送信号，但其自身会话中的进程除外，因此不会杀死其父shell
断电	告诉内核停止系统并关闭计算机（请参阅 停止 ）
重启	告诉内核重新引导系统（请参阅 暂停 ）
运行级别	报告上一个和当前运行级别，如上一个运行级别记录中所述 /var/run/utmp
关掉	以安全的方式关闭系统，向所有进程发出信号并通知所有已登录的用户
telinit	告诉 init 更改为哪个运行级别

6.77。Eudev-3.2.8

Eudev软件包包含用于动态创建设备节点的程序。

	预计编制
时间：	0.2 SBU
所需磁盘空间：	82 MB

6.77.1。Eudev的安装

准备Eudev进行编译：

```
./configure --prefix=/usr           \
            --bindir=/sbin          \
            --sbindir=/sbin         \
            --libdir=/usr/lib       \
            --sysconfdir=/etc       \
            --libexecdir=/lib       \
            --with-rootprefix=      \
            --with-rootlibdir=/lib  \
            --enable-manpages       \
            --disable-static
```

编译软件包：

```
make
```

现在创建测试所需的一些目录，但这些目录也将用作安装的一部分：

```
mkdir -pv /lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

要测试结果，请发出：

```
make check
```

安装软件包：

```
make install
```

安装一些在LFS环境中有用的自定义规则和支持文件：

```
tar -xvf ../udev-lfs-20171102.tar.xz
make -f udev-lfs-20171102/Makefile.lfs install
```

6.77.2。配置Eudev

有关硬件设备的信息在/etc/udev/hwdb.d和 /lib/udev/hwdb.d目录中维护 。 Eudev需要将该信息编译到二进制数据库中/etc/udev/hwdb.bin。创建初始数据库：

```
udevadm hwdb --update
```

每次更新硬件信息时都需要运行此命令。

6.77.3。Eudev的内容

已安装程序： udevadm和udevd
已安装的库： libudev.so
安装的目录： / etc / udev , / lib / udev和/ usr / share / doc / udev-udev-lfs-20171102

简短说明

udevadm	通用udev管理工具：控制udevd守护程序，从Udev数据库提供信息，监视uevent，等待uevent完成，测试Udev配置，并为给定设备触发uevent
udevd	一个守护程序，该守护程序在netlink套接字上侦听uevent，创建设备并运行配置的外部程序以响应这些uevent
libudev	udev设备信息的库接口
/etc/udev	包含Udev配置文件，设备权限和设备命名规则

6.78。关于调试符号

默认情况下，大多数程序和库都是使用调试符号（带有gcc的-g选项）进行编译的。这意味着在调试包含调试信息的已编译程序或库时，调试器不仅可以提供内存地址，还可以提供例程和变量的名称。

但是，这些调试符号的包含会大大扩展程序或库。以下是这些符号占用的空间量的示例：

- 甲的bash 二进制与调试符号：1200 KB
- 一个bash的 二进制不带调试符号：480 KB
- 带有调试符号的 Glibc和GCC文件（ /lib 和/usr/lib ）：87 MB
- 不含调试符号的Glibc和GCC文件：16 MB

大小可能会有所不同，具体取决于所使用的编译器和C库，但是在比较带有调试符号和不带有调试符号的程序时，差异通常是2到5之间。

因为大多数用户永远不会在其系统软件上使用调试器，所以通过删除这些符号可以重新获得大量磁盘空间。下一节将说明如何从程序和库中剥离所有调试符号。

6.79。再次剥离

本部分是可选的。如果目标用户不是程序员，并且不打算在系统软件上进行任何调试，则可以通过从二进制文件和库中删除调试符号来将系统大小减少约90 MB。除了不再完全调试软件之外，这不会造成任何不便。

使用下面提到的命令的大多数人都不会遇到任何困难。但是，很容易造成输入错误，并使新系统无法使用，因此在运行strip命令之前 ，最好将LFS系统备份为当前状态。

首先将所选库的调试符号放在单独的文件中。如果稍后在BLFS中运行使用 **valgrind**或 **gdb的**回归测试，则需要此调试信息。

```
save_lib="ld-2.30.so libc-2.30.so libpthread-2.30.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.6.0.27
             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
```

在执行剥离之前，请特别注意确保没有任何要剥离的二进制文件正在运行：

```
exec /tools/bin/bash
```

现在可以安全地剥离二进制文件和库了：

```
/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 {} ';'
```

大量文件将被报告为文件格式无法识别。可以安全地忽略这些警告。这些警告表明这些文件是脚本而不是二进制文件。

6.80。打扫干净

最后，清理运行测试遗留的一些额外文件：

```
rm -rf /tmp/*
```

现在注销，并使用更新的chroot命令重新进入chroot环境。从现在开始，在退出后需要重新进入chroot环境时，可以随时使用此更新的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/tools不再需要其中的程序。因此，您可以/tools 根据需要删除目录。

> **注意**
>
> 删除/tools还将删除用于运行工具链测试的Tcl，Expect和DejaGNU的临时副本。如果以后需要这些程序，则将需要重新编译并重新安装它们。BLFS书籍对此有说明（请参阅 **http://www.linuxfromscratch.org/blfs/** ）。

如果已通过手动方式或通过重新引导方式卸载了虚拟内核文件系统，请确保在重新输入chroot时已安装虚拟内核文件系统。第6.2.2节“安装和填充/ dev”和第6.2.3节“安装虚拟内核文件系统”中对此过程进行了说明。

为了满足多个软件包中的回归测试，本章前面有一些静态库没有被取消。这些库来自binutils，bzip2，e2fsprogs，flex，libtool和zlib。如果需要，请立即将其删除：

```
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
```

在/ usr / lib和/ usr / libexec目录中还安装了几个文件，文件名扩展名为.la。这些是“ libtool存档”文件，在Linux系统上通常是不需要的。此时，这些都没有必要。要删除它们，请运行：

```
find /usr/lib /usr/libexec -name \*.la -delete
```

有关libtool存档文件的更多信息，请参见 **BLFS部分“关于Libtool存档（.la）文件”**。

第7章系统配置

7.1。介绍

引导Linux系统涉及多项任务。该过程必须同时安装虚拟和实际文件系统，初始化设备，激活交换，检查文件系统的完整性，安装任何交换分区或文件，设置系统时钟，启动网络，启动系统所需的所有守护程序以及完成任何操作用户所需的其他自定义任务。必须组织此过程以确保任务以正确的顺序执行，但同时应尽快执行。

7.1.1。系统V

System V是经典的引导过程，大约从1983年开始就在Unix和类似Linux的系统中使用。它由一个小程序init组成，该程序设置基本程序，例如login（通过getty）并运行脚本。 。此脚本通常称为rc，控制一组附加脚本的执行，这些脚本执行初始化系统所需的任务。

该初始化 程序是由控制/etc/inittab文件，并整理成可以由用户运行的运行级别：

0-停止
1-单用户模式
2-多用户，不联网
3-完全多用户模式
4-用户可定义
5-带有显示管理器的完全多用户模式
6-重新启动

通常的默认运行级别是3或5。

优点

- 建立良好的系统。
- 易于定制。

缺点

- 慢启动。中速基本LFS系统需要8到12秒的时间，其中引导时间是从第一条内核消息到登录提示的时间。通常在登录提示后约2秒钟建立网络连接。
- 引导任务的串行处理。这与上一点有关。任何过程（例如文件系统检查）的延迟都将延迟整个引导过程。
- 不直接支持高级功能，例如控制组（cgroup）和每用户公平份额调度。
- 添加脚本需要手动的静态排序决策。

7.2。LFS-引导脚本-20190524

LFS-Bootscripts软件包包含一组脚本，用于在启动/关闭时启动/停止LFS系统。以下各节介绍了定制引导过程所需的配置文件和过程。

	预计编制
时间：	少于0.1 SBU
所需磁盘空间：	244 KB

7.2.1。安装LFS引导脚本

安装软件包：

```
make install
```

7.2.2。LFS引导脚本的内容

安装的脚本： checkfs，cleanfs，控制台，函数，停止，ifdown，ifup，localnet，模块，mountfs，mountvirtfs，网络，rc，重启，sendsignals，setclock，ipv4-static，swap，sysctl，sysklogd，模板，udev和udev_retry

安装目录： /etc/rc.d、/etc/init.d（符号链接），/ etc / sysconfig，/ lib / services，/ lib / lsb（符号链接）

简短说明

检查文件	在挂载之前检查文件系统的完整性（基于日志和基于网络的文件系统除外）
清洁剂	删除不应在重新启动之间保留的文件，例如/var/run/和中的文件/var/lock/；它重新创建 /var/run/utmp和删除可能存在的/etc/nologin，/fastboot和/forcefsck文件
安慰	为所需的键盘布局加载正确的键盘映射表；它还设置屏幕字体
功能	包含一些启动脚本使用的常用功能，例如错误和状态检查
停止	停止系统
ifdown	停止网络设备
ifup	初始化网络设备
本地网	设置系统的主机名和本地回送设备
模组	/etc/sysconfig/modules使用此处提供的参数 加载列出的内核模块
mountfs	挂载所有文件系统，标记为 *noauto* 或基于网络的 文件系统除外
mountvirtfs	挂载虚拟内核文件系统，例如 *proc*
网络	设置网络接口，例如网卡，并设置默认网关（如果适用）
rc	主运行级控制脚本；它负责按照要处理的符号链接的名称确定的顺序，依次运行所有其他引导脚本。
重启	重新启动系统
发送信号	确保在系统重新引导或停止之前终止每个进程
时钟	如果硬件时钟未设置为UTC时间，则将内核时钟重置为本地时间
ipv4-static	提供将静态Internet协议（IP）地址分配给网络接口所需的功能
交换	启用和禁用交换文件和分区
系统控制	将系统配置值/etc/sysctl.conf（如果存在）从正在运行的内核中加载
sysklogd	启动和停止系统和内核日志守护程序
模板	用于为其他守护程序创建自定义启动脚本的模板

乌德夫	准备/dev 目录并启动Udev
udev_retry	重试失败的udev uevents，并根据需要将 生成的规则文件从复制/run/udev到/etc/udev/rules.d

7.3。设备和模块处理概述

在第6章中，我们在构建eudev时安装了Udev软件包。在详细介绍其工作原理之前，请先简要介绍一下以前处理设备的方法。

通常，Linux系统通常使用静态设备创建方法，通过这种方法，可以在下面创建大量设备节点/dev（有时实际上是数千个节点），而不管是否实际存在相应的硬件设备。这通常是通过MAKEDEV脚本完成的，该脚本包含对mknod程序的多次调用，其中包含世界上可能存在的每种可能设备的相关主要和次要设备编号。

使用Udev方法，只有内核检测到的那些设备才能获得为它们创建的设备节点。由于这些设备节点将在每次系统引导时创建，因此它们将存储在*devtmpfs* 文件系统（完全位于系统内存中的虚拟文件系统）上。设备节点不需要太多空间，因此使用的内存可以忽略不计。

7.3.1。历史

在2000年2月，一个新的文件系统*devfs*被合并到2.3.46内核中，并在2.4系列稳定内核中可用。尽管它存在于内核源代码本身中，但这种动态创建设备的方法从未获得核心内核开发人员的压倒性支持。

采用的方法的主要问题*devfs*是它处理设备检测，创建和命名的方式。后一个问题，即设备节点命名问题，可能是最关键的。通常接受的是，如果允许设备名称是可配置的，则设备命名策略应由系统管理员决定，而不是由任何特定开发人员强加给他们。的*devfs*文件系统还遭受设计中固有的竞争条件，如果不对内核进行实质性修订，就无法修复。由于缺乏维护，它已被标记为长期不推荐使用，并于2006年6月最终从内核中删除。

随着不稳定的2.5内核树的发展（后来作为稳定系列2.6系列发布），一个新的虚拟文件系统*sysfs*应运而生。的工作*sysfs*是将系统硬件配置的视图导出到用户空间进程。通过这种对用户空间可见的表示，开发替代用户空间的可能性*devfs*变得更加现实。

7.3.2。Udev实施

7.3.2.1。Sysfs

该*sysfs*文件系统上面简要提及。也许您会想知道如何*sysfs*知道系统上存在的设备以及应该使用哪些设备编号。内核*sysfs*检测到驱动程序后，已编译到内核中的驱动程序会直接向其注册对象（内部为devtmpfs）。对于编译为模块的驱动程序，将在模块加载时进行此注册。一旦*sysfs*文件系统挂载（在/ sys上），驱动程序就向其注册的数据 *sysfs* 可用于用户空间进程和udevd进行处理（包括对设备节点的修改）。

7.3.2.2。设备节点创建

设备文件由内核由*devtmpfs*文件系统创建。任何希望注册设备节点*devtmpfs*的驱动程序都会 （通过驱动程序核心）进行操作。当*devtmpfs*实例安装在上时 /dev，将首先使用固定的名称，权限和所有者创建设备节点。

不久之后，内核将向udevd发送一个uevent 。基于在内的文件中指定的规则/etc/udev/rules.d，/lib/udev/rules.d和/run/udev/rules.d目录， udevd会将创建的符号链接附加到该设备节点，或改变它的权限，拥有者，或基，或修改内部 的udevd 该对象数据库条目（名称）。

这三个目录中的规则编号，并且所有三个目录合并在一起。如果udevd找不到正在创建的设备的规则，它将把权限和所有权保留为*devtmpfs*最初使用的内容。

7.3.2.3。模块加载

编译为模块的设备驱动程序可能内置有别名。别名在modinfo 程序的输出中可见， 通常与模块支持的设备的总线特定标识符有关。例如，*snd-fm801*驱动程序支持供应商ID为0x1319且设备ID为0x0801的PCI设备，并且别名为" pci：v00001319d00000801sv * sd * bc04sc01i * "。对于大多数设备，总线驱动程序通过导出将处理设备的驱动程序的别名*sysfs*。例如，/sys/bus/pci/devices/0000:00:0d.0/modalias 文件可能包含字符串" pci：v00001319d00000801sv00001319sd00001319bc04sc01i00 "。Udev提供的默认规则将使 udevd使用uevent环境变量的内容（应与sysfs 中的文件的内容相同）调出/ sbin / modprobe，从而加载其别名与该字符串匹配的所有模块通配符扩展后。 MODALIASmodalias

在这个例子中，这意味着，除了 *SND-FM801*，废弃（和不想要的）*的长处*的驱动程序将被加载，如果它是可用的。请参阅以下有关防止不必要的驱动程序加载的方法。

内核本身也能够按需加载用于网络协议，文件系统和NLS支持的模块。

7.3.2.4。处理可热插拔/动态设备

当您插入通用串行总线（USB）MP3播放器之类的设备时，内核会识别出该设备现在已连接并生成一个uevent。然后，如上所述，由udevd处理该事件。

7.3.3。加载模块和创建设备时出现问题

自动创建设备节点时可能会遇到一些问题。

7.3.3.1。内核模块不会自动加载

Udev仅在具有特定于总线的别名并且总线驱动程序正确将必需的别名导出到的情况下才会加载模块*sysfs*。在其他情况下，应通过其他方式安排模块加载。使用Linux-5.2.8，已知Udev会为INPUT，IDE，PCI，USB，SCSI，SERIO和FireWire设备加载正确编写的驱动程序。

要确定所需的设备驱动程序是否具有对Udev的必要支持，请以模块名称作为参数运行modinfo。现在尝试在下面找到设备目录，/sys/bus并检查那里是否有modalias 文件。

如果该modalias文件位于中*sysfs*，则驱动程序支持该设备并可以直接与其通信，但没有别名，这是驱动程序中的错误。在没有Udev帮助的情况下加载驱动程序，并希望稍后解决此问题。

如果modalias下方的相关目录中没有文件，则/sys/bus意味着内核开发人员尚未向该总线类型添加对模式的支持。对于Linux-5.2.8，ISA总线就是这种情况。期望此问题在更高版本的内核中得以解决。

Udev根本不打算加载" 包装 "驱动程序，例如 *snd-pcm-oss*和非硬件驱动程序，例如*循环*。

7.3.3.2。内核模块不会自动加载，而Udev并不打算加载它

如果" 包装器 "模块仅增强了某些其他模块提供的功能（例如， *snd-pcm-oss*通过使声卡可用于OSS应用程序增强了*snd-pcm*的功能），请配置 modprobe以在Udev加载后加载包装器包装的模块。为此，在相应 文件中添加" softdep "行。例如： /etc/modprobe.d/*<filename>*.conf

```
softdep snd-pcm post: snd-pcm-oss
```

注意，" softdep "命令也允许 pre:依赖关系，或两者的混合物pre:和 post:。有关" softdep "语法和功能的modprobe.d(5)更多信息，请参见手册页。

如果有问题的模块不是包装器，并且本身是有用的，请配置模块引导脚本以在系统引导时加载该模块。为此，请/etc/sysconfig/modules在单独的行中将模块名称添加到文件中。这也适用于包装器模块，但是在这种情况下不是最优的。

7.3.3.3。Udev加载了一些不需要的模块

要么不构建模块，要么不像 下面示例中/etc/modprobe.d/blacklist.conf的*forte*模块那样在文件中将其列入黑名单 ：

```
blacklist forte
```

仍然可以使用显式modprobe命令手动加载列入黑名单的模块。

7.3.3.4。Udev错误地创建设备，或建立错误的符号链接

如果规则意外匹配设备，通常会发生这种情况。例如，写得不好的规则可能会按供应商匹配SCSI磁盘（根据需要）和相应的SCSI通用设备（错误地）。在udevadm info命令的帮助下，找到有问题的规则并将其更具体。

7.3.3.5。Udev规则运行不可靠

这可能是先前问题的另一个体现。如果不是，并且您的规则使用*sysfs*属性，则可能是内核计时问题，需要在以后的内核中解决。现在，您可以通过创建一条等待使用的*sysfs*属性的规则并将其附加到/etc/udev/rules.d/10-wait_for_sysfs.rules 文件中来解决该问题（如果此文件不存在，请创建该文件）。如果这样做，请通知LFS开发列表，它会有所帮助。

7.3.3.6。Udev不创建设备

进一步的文字假设驱动程序是静态内置在内核中的，或者已经作为模块加载，并且您已经检查了Udev不会创建错误命名的设备。

如果内核驱动程序不将其数据导出到，则Udev不需要创建设备节点的信息*sysfs*。这是内核树外部的第三方驱动程序最常见的情况。 /lib/udev/devices使用适当的主/次编号创建一个静态设备节点（请参阅devices.txt内核文档中的文件或第三方驱动程序供应商提供的文档）。静态设备节点将被复制到/dev由udev的。

7.3.3.7。重新启动后，设备命名顺序会随机更改

这是由于Udev在设计上可以并行处理uevents并加载模块，因此顺序不可预测。这将永远不会是" 固定的 "。您不应该依赖稳定的内核设备名称。取而代之的是，根据设备的某些稳定属性（例如序列号或Udev安装的各种* _id实用程序的输出），创建自己的规则来使用稳定名称进行符号链接。有关示例，请参见第7.4节"管理设备"和第7.5节"常规网络配置"。

7.3.4。有用的阅读

以下站点提供了其他有用的文档：

- **http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf** 的用户空间实现*devfs*

- 该*sysfs*文件系统 http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf

7.4。管理设备

7.4.1。网络设备

默认情况下，Udev根据固件/ BIOS数据或物理特性（如总线，插槽或MAC地址）来命名网络设备。该命名约定的目的是确保网络设备的命名一致，而不是基于发现网卡的时间。例如，在具有两个由Intel和Realtek制造的网卡的计算机上，由Intel制造的网卡可能会变成eth0，而Realtek卡会变成eth1。在某些情况下，重新启动后，卡会以其他方式重新编号。

在新的命名方案中，典型的网络设备名称将类似于enp5s0或wlp3s0。如果不需要此命名约定，则可以实现传统的命名方案或自定义方案。

7.4.1.1。在内核命令行上禁用持久命名

可以通过net.ifnames=0在内核命令行上添加来恢复使用eth0，eth1等的传统命名方案。这对于仅具有一个相同类型的以太网设备的系统最为合适。便携式计算机通常具有多个名为eth0和wlan0的以太网连接，它们也是该方法的候选对象。命令行在GRUB配置文件中传递。请参见第8.4.4节“创建GRUB配置文件”。

7.4.1.2。创建自定义Udev规则

可以通过创建自定义Udev规则来自定义命名方案。已包含一个脚本，该脚本生成初始规则。通过运行以下规则来生成这些规则：

```
bash /lib/udev/init-net-rules.sh
```

现在，检查/etc/udev/rules.d/70-persistent-net.rules 文件，找出分配给哪个网络设备的名称：

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

> **注意**
>
> 在某些情况下，例如当MAC地址已经手动分配给网卡或在虚拟环境（例如Qemu或Xen）中分配给网络卡时，可能不会生成网络规则文件，因为地址分配不一致。在这些情况下，无法使用此方法。

该文件以注释块开头，后跟每个NIC的两行。每个NIC的第一行是带注释的说明，显示其硬件ID（例如，如果是PCI卡，则为PCI供应商和设备ID），如果可以找到驱动程序，则在括号中加上驱动程序。硬件ID和驱动程序都不能用来确定接口的名称。此信息仅供参考。第二行是与该NIC匹配并实际上为其分配名称的Udev规则。

所有Udev规则均由多个键组成，并以逗号和可选的空格分隔。该规则的键以及每个键的说明如下：

- SUBSYSTEM=="net" -这告诉Udev忽略不是网卡的设备。

- ACTION=="add" -这告诉Udev对不是添加项的uevent忽略该规则（“ remove”和“ change” uevents也会发生，但不需要重命名网络接口）。
- DRIVERS=="?*"-存在的原因是Udev将忽略VLAN或网桥子接口（因为这些子接口没有驱动程序）。跳过这些子接口，因为要分配的名称将与其父设备冲突。
- ATTR{address} -此项的值是NIC的MAC地址。
- ATTR{type}=="1"-这样可以确保该规则仅在某些无线驱动程序的情况下匹配主接口，从而创建多个虚拟接口。跳过辅助接口的原因与跳过VLAN和网桥子接口的原因相同：否则将发生名称冲突。
- NAME -此项的值是Udev将分配给该接口的名称。

的价值NAME是重要的部分。在继续操作之前，请确保您已为每个网卡分配了哪个名称，并在NAME 下面创建配置文件时确保使用该值。

7.4.2。CD-ROM符号链接

您稍后可能希望安装的某些软件（例如，各种媒体播放器）期望/dev/cdrom和/dev/dvd符号链接存在，并指向CD-ROM或DVD-ROM设备。另外，将对这些符号链接的引用放入中可能会很方便/etc/fstab。Udev带有一个脚本，该脚本将根据每个设备的功能生成规则文件来为您创建这些符号链接，但是您需要确定希望使用脚本的两种操作模式中的哪一种。

首先，脚本可以在“ 按路径 ”模式下运行（默认用于USB和FireWire设备），该脚本创建的规则取决于CD或DVD设备的物理路径。其次，它可以在“ by-id ”模式下运行（IDE和SCSI设备的默认设置），在此模式下，它创建的规则取决于CD或DVD设备本身中存储的标识字符串。路径由Udev的path_id脚本确定，标识字符串由其ata_id或 scsi_id从硬件中读取 程序，具体取决于您拥有的设备类型。

每种方法都有其优点；正确的使用方法将取决于可能发生的设备更改类型。如果您希望更改设备的物理路径（即设备插入的端口和/或插槽），例如因为计划将驱动器移动到其他IDE端口或其他USB连接器，则应该使用“ by-id ”模式。另一方面，如果您希望设备的标识发生变化（例如因为它可能会失效），并且您将其替换为具有相同功能且插入相同连接器的其他设备，则应使用 “ by -path ”模式。

如果驱动器可能发生两种类型的更改，请根据您希望更频繁发生的更改类型选择一种模式。

> **重要**
>
> 外部设备（例如，连接USB的CD驱动器）不应使用按路径持久性，因为每次将设备插入新的外部端口时，其物理路径都会改变。如果您编写Udev规则以通过其物理路径识别它们，则所有与外部连接的设备都将出现此问题。该问题不仅限于CD和DVD驱动器。

如果您希望查看Udev脚本将使用的值，则对于适当的CD-ROM设备，在/sys（例如，可以是/sys/block/hdd）下找到相应的目录，然后运行类似于以下命令：

```
udevadm test /sys/block/hdd
```

查看包含各种* _id程序输出的行。该“ 通过-ID ”，如果它存在，并且不为空模式将使用ID_SERIAL值，否则将使用ID_MODEL和ID_REVISION的组合。该 “ 通过路径 ”模式将使用ID_PATH值。

如果默认模式不适合您的情况，那么下面的修改可以进行到/etc/udev/rules.d/83-cdrom-symlinks.rules 文件，如下（其中*mode*是一个“ 由-ID ” 或“ 由路径 ”）：

```
sed -i -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    /etc/udev/rules.d/83-cdrom-symlinks.rules
```

请注意，此时不必创建规则文件或符号链接，因为您已将主机/dev目录绑定安装到LFS系统中，并且我们假定符号链接位于主机上。规则和符号链接将在您首次引导LFS系统时创建。

但是，如果有多个CD-ROM设备，则此时生成的符号链接可能指向的设备与主机上指向的设备不同，这是因为未按可预测的顺序发现设备。首次引导LFS系统时创建的分配将保持稳定，因此仅当您需要两个系统上的符号链接指向同一设备时，这才是问题。如果需要，请/etc/udev/rules.d/70-persistent-cd.rules 在引导后检查（并可能编辑）生成的文件，以确保分配的符号链接与所需内容相匹配。

7.4.3。处理重复的设备

如第7.3节"设备和模块处理概述"中所述，具有相同功能的设备出现的顺序 /dev实质上是随机的。例如，如果您有USB网络摄像头和电视调谐器，则有时 /dev/video0指的是照相机，而/dev/video1指的是调谐器，有时在重新引导后，顺序更改为相反的顺序。对于除声卡和网卡以外的所有类别的硬件，此问题可以通过为自定义持久性符号链接创建Udev规则来解决。网卡的情况在可以在 **BLFS中**找到第7.5节"常规网络配置"和声卡配置。

对于每个可能出现此问题的设备（即使当前Linux发行版中不存在该问题），也可以在/sys/class或下找到相应的目录 /sys/block。对于视频设备，可能是 。找出唯一标识设备的属性（通常，供应商和产品ID和/或序列号有效）： /sys/class/video4linux/video*X*

```
udevadm info -a -p /sys/class/video4linux/video0
```

然后编写创建符号链接的规则，例如：

```
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
```

其结果是，/dev/video0 和/dev/video1设备仍然随机指向调谐器和网络摄像机（并且因此不应该被直接使用），但是符号链接 /dev/tvtuner和/dev/webcam总是指向正确的设备。

7.5。常规网络配置

7.5.1。创建网络接口配置文件

网络脚本打开和关闭哪些接口通常取决于中的文件/etc/sysconfig/。该目录应包含要配置的每个接口的文件，例如 ifconfig.xyz，其中“ xyz ”应描述网卡。接口名称（例如eth0）通常是合适的。该文件内部是该接口的属性，例如其IP地址，子网掩码等。文件名的茎必须为*ifconfig*。

> **注意**
>
> 如果未使用上一节中的过程，则Udev将根据系统物理特征（例如enp2s1）分配网卡接口名称。如果不确定接口名称是什么，则在引导系统后始终可以运行 ip link或 ls / sys / class / net。

以下命令为具有静态IP地址的*eth0*设备创建示例文件 ：

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

必须在每个文件中更改斜体值以匹配正确的设置。

如果将该ONBOOT变量设置为“ yes ”，则系统V网络脚本将在系统引导期间启动网络接口卡（NIC）。如果设置为“ yes ”以外 的任何值，则网络脚本将忽略NIC，并且不会自动将其启动。可以使用ifup和ifdown命令手动启动或停止该接口。

该IFACE变量定义接口名称，例如eth0。所有网络设备配置文件都需要此文件。文件扩展名必须与此值匹配。

该SERVICE变量定义用于获取IP地址的方法。LFS-Bootscripts软件包具有模块化IP分配格式，并且在/lib/services/目录中创建其他文件可以使用其他IP分配方法。这通常用于动态主机配置协议（DHCP），这在BLFS手册中已解决。

该GATEWAY变量应包含默认网关IP地址（如果存在）。如果不是，则完全注释掉变量。

该PREFIX变量包含子网中使用的位数。IP地址中的每个八位位组均为8位。如果子网的网络掩码是255.255.255.0，则它使用前三个八位位组（24位）来指定网络号。如果网络掩码为255.255.255.240，则它将使用前28位。DSL和基于电缆的Internet服务提供商（ISP）通常使用长度超过24位的前缀。在此示例（PREFIX = 24）中，网络掩码为255.255.255.0。PREFIX根据您的特定子网调整变量。如果省略，则PREFIX默认为24。

有关更多信息，请参见ifup手册页。

7.5.2。创建/etc/resolv.conf文件

系统将需要某种方式来获取域名服务（DNS）名称解析，以将Internet域名解析为IP地址，反之亦然。最好将ISP或网络管理员提供的DNS服务器的IP地址放在中，以达到最佳效果/etc/resolv.conf。通过运行以下命令创建文件：

```
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
```

该domain语句可以省略，也可以替换为search语句。有关更多详细信息，请参见手册页中的resolv.conf。

替换*<IP address of the nameserver>*为最适合设置的DNS的IP地址。通常会有一个以上的条目（需求要求辅助服务器具有后备功能）。如果只需要或想要一个DNS服务器，请 从文件中删除第二个*名称服务器*行。该IP地址也可以是本地网络上的路由器。

> **注意**
>
> Google公用IPv4 DNS地址为8.8.8.8和8.8.4.4。

7.5.3。配置系统主机名

在引导过程中，该文件/etc/hostname用于建立系统的主机名。

创建/etc/hostname文件并通过运行以下命令输入主机名：

```
echo "<lfs>" > /etc/hostname
```

*<lfs>*需要替换为计算机提供的名称。不要在此处输入完全合格的域名（FQDN）。该信息被放入/etc/hosts文件中。

7.5.4。定制/ etc / hosts文件

确定IP地址，标准域名（FQDN）和/etc/hosts文件中可能使用的别名。语法为：

```
IP_address myhost.example.org aliases
```

除非计算机对Internet可见（即，有一个注册域和一个有效的分配IP地址块-大多数用户没有此地址），否则请确保IP地址在专用网络IP地址范围内。有效范围是：

```
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范围内的任何数字。y可以是0-255之间的任何数字。

有效的专用IP地址可以是192.168.1.1。该IP的有效FQDN可能是lfs.example.org。

即使不使用网卡，仍然需要有效的FQDN。这对于某些程序正确运行是必需的。

/etc/hosts通过运行以下命令 创建文件：

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

的*<192.168.1.1>*， *<FQDN>*和 *<HOSTNAME>* 值需要改变用于特定用途或要求（如果由网络/系统管理员分配一个IP地址和机器将被连接到现有的网络）。可选的别名可以省略。

7.6。System V引导脚本的使用和配置

7.6.1。System V引导脚本如何工作？

Linux使用一种名为SysVinit的特殊启动工具，该工具基于*运行级别*的概念。从一个系统到另一个系统，它可能有很大的不同，因此不能假定由于事情在一个特定的Linux发行版中可以工作，因此它们在LFS中也应相同。LFS有其自己的处理方式，但它遵循公认的标准。

SysVinit（ 从现在开始将称为“ init ”）使用运行级别方案进行工作。有七个运行级别（从0到6编号）（实际上，运行级别更多，但是它们是针对特殊情况的，通常不使用。有关init(8)更多详细信息，请参见。 ），每个运行级别对应于计算机应该在启动时执行。缺省运行级别是3。这是实施时不同运行级别的描述：

0：停止计算机
1：单用户模式
2：无网络的多用户模式
3：有网络的多用户模式
4：保留用于自定义，否则与3

相同5：与4相同，通常用于GUI登录（例如X的 xdm 或KDE的 kdm）
6：重新启动计算机

7.6.2。配置Sysvinit

在内核初始化期间，运行的第一个程序是在命令行上指定的，或者默认是 init。该程序读取初始化文件/etc/inittab。使用以下命令创建此文件：

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

id:3:initdefault:

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

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

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

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

有关初始化文件的说明在*inittab*的手册页中。对于LFS，运行的关键命令是rc。上面的初始化文件将指示rc运行所有脚本，该脚本以/etc/rc.d/rcS.d目录中的S开头，然后以/etc/rc.d/rc?.dinitdefault值指定问号的目录中的所有以S开头的脚本运行。

为方便起见，rc脚本读取中的函数库/lib/lsb/init-functions。该库还读取可选的配置文件/etc/sysconfig/rc.site。后续小节中描述的任何系统配置文件参数都可以替换地放置在此文件中，以允许将所有系统参数合并到该文件中。

为了方便调试，功能脚本还将所有输出记录到/run/var/bootlog。由于该/run目录是tmpfs，因此该文件在引导过程中不是持久性的，但是/var/log/boot.log在引导过程结束时会附加到更永久的文件中。

7.6.2.1。更改运行级别

改变运行级别与完成初始化*<runlevel>*，其中*<runlevel>*是目标运行级别。例如，要重新启动计算机，用户可以发出init 6命令，该命令是重新启动 命令的别名 。同样，init 0是halt命令的别名。

有下一批目录的/etc/rc.d那个样子rc?.d（这里？是运行级别的数量）和rcsysinit.d所有包含的符号链接。有些以*K*开头，另一些以*S*开头，并且所有字母的首字母后都有两个数字。K表示停止（终止）服务，S表示开始服务。数字确定脚本的运行顺序，从00到99-数字越低，脚本执行得越早。当初始化如果切换到另一个运行级别，则根据选择的运行级别启动或停止适当的服务。

真正的脚本在中/etc/rc.d/init.d。它们完成实际工作，并且符号链接都指向它们。K个链接和S个链接指向中的相同脚本/etc/rc.d/init.d。这是因为该脚本可以调用不同的参数一样 *start*，*stop*，*restart*，*reload*，和*status*。遇到K链接时，将使用*stop*参数运行适当的脚本 。遇到S链接时，将使用*start* 参数运行适当的脚本。

这种解释有一个例外。在和目录中以*S*开头的链接不会导致任何启动。将使用参数 来停止它们。其背后的逻辑是，当用户要重新引导或暂停系统时，无需启动任何操作。只需要停止系统。 rc0.drc6.d*stop*

这些是参数使脚本执行的描述：

start
服务已启动。

stop
服务已停止。

restart
该服务已停止，然后再次启动。

reload
服务的配置已更新。当不需要重新启动服务时，在修改服务的配置文件后使用此属性。

status
告知服务是否正在运行以及使用哪些PID。

随意修改引导过程的工作方式（毕竟，这是您自己的LFS系统）。此处给出的文件是如何完成此操作的示例。

7.6.3。Udev引导脚本

该/etc/rc.d/init.d/udev initscript启动udevd，触发内核已经创建的所有“ coldplug”设备，并等待任何规则完成。该脚本还将默认的取消设置uevent处理程序/sbin/hotplug。这样做是因为内核不再需要调用外部二进制文件。相反，udevd 将在netlink套接字上侦听内核引发的uevent。

该/etc/rc.d/init.d/udev_retry 启动脚本需要照顾的子系统，其规则可能依赖于未安装，直到文件系统重新触发事件mountfs脚本的运行（特别是，/usr和 /var可能导致此）。该脚本在mountfs脚本之后运行，因此这些规则（如果重新触发）应该第二次成功。它是从/etc/sysconfig/udev_retry文件配置的；该文件中除注释以外的任何单词均视为在重试时触发的子系统名称。要查找设备的子系统，请使用udevadm info --attribute-walk <设备>，其中<设备>是/ dev或/ sys中的绝对路径，例如/ dev / sr0或/ sys / class / rtc。

有关内核模块加载和udev的信息，请参见 第7.3.2.3节“模块加载”。

7.6.4。配置系统时钟

所述setclock 脚本读取从硬件时钟，也称为BIOS或互补金属氧化物半导体（CMOS）时钟的时间。如果硬件时钟设置为UTC，则此脚本将使用/etc/localtime 文件（告诉hwclock程序用户所在的时区）将硬件时钟的时间转换为本地时间。无法检测硬件时钟是否设置为UTC，因此需要手动配置。

当内核在启动时检测到硬件功能时 ，setclock通过udev运行。也可以使用stop参数手动运行它，以将系统时间存储到CMOS时钟。

如果您不记得硬件时钟是否设置为UTC，请通过运行hwclock --localtime --show 命令进行查找。这将根据硬件时钟显示当前时间。如果此时间与您的手表说的相符，则硬件时钟将设置为本地时间。如果hwclock的输出不是本地时间，则很有可能将其设置为UTC时间。通过在hwclock所示的时间上加上或减去时区的适当小时数来验证这一点。例如，如果您当前处于MST时区（也称为GMT -0700），则将本地时间增加7个小时。

如果硬件时钟*未*设置为UTC时间 ，则将UTC 下面的变量的值更改为*0*（零）。

/etc/sysconfig/clock通过运行以下 命令创建一个新文件：

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

UTC=1

# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=

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

http://www.linuxfromscratch.org/hints/downloads/files/time.txt 提供了一个很好的提示，说明了如何处理LFS上的时间。它解释了诸如时区，UTC和 TZ环境变量之类的问题。

> **注意**
>
> 可以在/etc/sysconfig/rc.site 文件中另外设置CLOCKPARAMS和UTC参数。

7.6.5。配置Linux控制台

本节讨论如何配置控制台启动脚本，以设置键盘映射，控制台字体和控制台内核日志级别。如果将不使用非ASCII字符（例如，版权符号，英镑符号和欧元符号）并且键盘为美式键盘，则可以跳过本节的大部分内容。没有配置文件（或中的等效设置 rc.site），控制台引导脚本将不执行任何操作。

在控制台 脚本读取/etc/sysconfig/console的配置信息文件。确定将使用哪种键盘映射和屏幕字体。各种特定于语言的HOWTO也可以提供帮助，请参见 **http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html**。如果仍然不确定，请在/usr/share/keymaps和/usr/share/consolefonts目录中查找有效的键

盘映射和屏幕字体。阅读loadkeys(1)和setfont(8)手册页以确定这些程序的正确参数。

该/etc/sysconfig/console文件应包含以下格式的行：VARIABLE =" value"。可以识别以下变量：

逻辑水平
此变量指定dmesg设置的发送到控制台的内核消息的日志级别。有效级别是从" 1"（无消息）到" 8"。默认级别为" 7"。

键盘映射
此变量指定loadkeys 程序的参数 ，通常是要加载的键映射的名称，例如 " it "。如果未设置此变量，则引导脚本将不会运行loadkeys程序，将使用默认的内核键盘映射。请注意，一些按键映射具有相同名称的多个版本（cz及其变体在qwerty /和qwertz /中，es在olpc /和qwerty /中，而trf在fgGIod /和qwerty /中）。在这些情况下，还应指定父目录（例如qwerty / es），以确保加载正确的键盘映射。

KEYMAP_CORRECTIONS
这个（很少使用的）变量指定第二次调用loadkeys程序的参数。如果库存键图不能完全令人满意并且必须进行少量调整，则这很有用。例如，要将欧元符号包括在通常没有它的键盘图中，请将此变量设置为 " euro2 "。

字体
此变量指定setfont 程序的参数 。通常，这包括字体名称 " -m "和要加载的应用程序字符映射的名称。例如，为了将" lat1-16 "字体与" 8859-1 "应用程序字符映射一起加载（在美国适当），请将此变量设置为" lat1-16 -m 8859-1 "。在UTF-8模式下，内核使用应用程序字符映射将键映射中的8位组合键代码转换为UTF-8，因此应将" -m"参数的参数设置为组合字符的编码。键映射中的键代码。

统一码
将此变量设置为" 1 "，" yes "或" true "，以使控制台进入UTF-8模式。这在基于UTF-8的语言环境中很有用，否则有害。

LEGACY_CHARSET
对于许多键盘布局，Kbd软件包中没有现货的Unicode键映射。该控制台初始化脚本不会如果该变量设置为可用非UTF-8键盘映射的编码可用键盘映射转换为UTF-8的飞行。

一些例子：

- 对于非Unicode设置，通常只需要KEYMAP和FONT变量。例如，对于波兰语设置，将使用：

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

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- 如上所述，有时有必要稍微调整库存键图。下面的示例将欧元符号添加到德语键盘映射：

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

KEYMAP="de-latin1"
```

```
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# End /etc/sysconfig/console
EOF
```

- 以下是保加利亚语的支持Unicode的示例，其中存在库存的UTF-8键盘映射：

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

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

- 由于在上一个示例中使用了512字形的LatArCyrHeb-16字体，因此，除非使用帧缓冲区，否则Linux控制台将不再提供亮色。如果想在没有帧缓冲区的情况下保持鲜艳的色彩并且可以在没有不属于其语言的字符的情况下生存，那么仍然可以使用特定于语言的256字形字体，如下所示：

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

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- 以下示例说明了从ISO-8859-15到UTF-8的键映射自动转换以及在Unicode模式下启用死键：

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

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"
```

```
# End /etc/sysconfig/console
EOF
```

- 一些键盘映射具有死键（即，键本身不会产生字符，而是在下一个键产生的字符上加重音）或定义合成规则（例如：“ 按Ctrl +。AE获得Æ ”）在默认键盘图中）。仅当要组合在一起的源字符不是多字节时，Linux-5.2.8才能正确解释键图中的死键和组合规则。这种缺陷不会影响欧洲语言的键盘映射，因为在非重音ASCII字符上添加了重音，或者两个ASCII字符被组合在一起。但是，在UTF-8模式下，这是一个问题，例如，对于希腊语来说，有时需要在字母“ alpha ”上加重音 。解决方案是避免使用UTF-8，或者安装在输入处理中没有此限制的X窗口系统。
- 对于中文，日文，韩文和其他一些语言，不能将Linux控制台配置为显示所需的字符。需要这些语言的用户应安装X Window系统，覆盖必要字符范围的字体以及正确的输入法（例如，SCIM，它支持多种语言）。

> **注意**
>
> 该/etc/sysconfig/console 文件仅控制Linux文本控制台的本地化。它与通过ssh会话或串行控制台在X Window系统中设置正确的键盘布局和终端字体无关。在这种情况下，以上最后两个列表项中提到的限制将不适用。

7.6.6。在启动时创建文件

有时，需要在引导时创建文件。例如，/tmp/.ICE-unix 可能需要目录。这可以通过在/etc/sysconfig/createfiles配置脚本中创建一个条目来完成。该文件的格式嵌入在默认配置文件的注释中。

7.6.7。配置sysklogd脚本

该sysklogd脚本将syslogd程序作为System V初始化的一部分进行调用 。该-m 0选项默认关闭syslogd每20分钟写入一次日志文件的定期时间戳记。如果要打开此定期时间戳记，请编辑/etc/sysconfig/rc.site变量SYSKLOGD_PARMS并将其定义为所需的值。例如，要删除所有参数，请将变量设置为空值：

```
SYSKLOGD_PARMS =
```

请参阅man syslogd以获取更多选项。

7.6.8。rc.site文件

可选/etc/sysconfig/rc.site文件包含为每个SystemV启动脚本自动设置的设置。它可以交替设置在指定的值 hostname ，console以及clock在文件/etc/sysconfig/目录。如果在这两个单独的文件和中都存在关联的变量rc.site ，则脚本特定文件中的值优先。

rc.site还包含可以自定义引导过程其他方面的参数。设置IPROMPT变量将启用引导脚本的选择性运行。其他选项在文件注释中描述。该文件的默认版本如下：

```
#rc.site
#引导脚本的可选参数。

#发行信息
```

```
#这些值（如果在此处指定）将覆盖默认值
#DISTRO =“ Linux From Scratch”#发行名称
#DISTRO_CONTACT =“ lfs-dev@linuxfromscratch.org”#错误报告地址
#DISTRO_MINI =“ LFS”#在发行版配置的文件名中使用的短名称

#定义在打印到屏幕上的消息中使用的自定义颜色

#请查阅`man console_codes`了解更多信息
#在“ ECMA-48设置图形渲染”部分下
#
#警告：从8位字体切换到9位字体时，
#linux控制台将把粗体（1;）重新解释为
#9bit字体的前256个字形。这确实
#不影响帧缓冲控制台

#这些值（如果在此处指定）将覆盖默认值
#BRACKET =“ \\ 033 [1; 34m”#蓝色
#FAILURE =“ \\ 033 [1; 31m”#红色
#INFO =“ \\ 033 [1; 36m”#青色
#NORMAL =“ \\ 033 [0; 39m”#灰色
#SUCCESS =“ \\ 033 [1; 32m”#绿色
#WARNING =“ \\ 033 [1; 33m”#黄色

#使用彩色前缀
#这些值（如果在此处指定）将覆盖默认值
#BMPREFIX =“”
#SUCCESS_PREFIX =“ $ {SUCCESS} * $ {NORMAL}”
#FAILURE_PREFIX =“ $ {FAILURE} ***** $ {NORMAL}”
#WARNING_PREFIX =“ $ {WARNING} *** $ {NORMAL}”

#手动看到消息输出的右边缘（字符）
#在引导期间重置控制台字体以覆盖时很有用
#自动屏幕宽度检测
#COLUMNS = 120

#互动启动
#IPROMPT =“是”#是否显示交互式启动提示
#itime =“ 3”#显示提示的时间（以秒为单位）

#发行版欢迎字符串的总长度，不包含转义码
#wlen = $（回显“欢迎使用$ {DISTRO}” | wc -c）
#welcome_message =“欢迎使用$ {INFO} $ {DISTRO} $ {NORMAL}”

#交互字符串的总长度，不包含转义码
#ilen = $（回显“按'I'进入交互式启动” | wc -c）
#i_message =“按'$ {FAILURE} I $ {NORMAL}'进入交互式启动”
```

```
#设置脚本以在重新引导时跳过文件系统检查
#FASTBOOT =是

#从控制台跳过阅读
#HEADLESS =是

#如果是，写出fsck进度
#VERBOSE_FSCK =否

#加快启动速度，而无需等待udev解决
#OMIT_UDEV_SETTLE = y

#加快启动速度，而无需等待udev_retry的解决
#OMIT_UDEV_RETRY_SETTLE =是

#如果是，请跳过清洁/ tmp
#SKIPTMPCLEAN =否

#对于setclock
#UTC = 1
#CLOCKPARAMS =

#对于consolelog（请注意，默认值7 = debug嘈杂）
#LOGLEVEL = 7

#对于网络
#HOSTNAME = mylfs

#关闭时TERM和KILL信号之间的延迟
#KILLDELAY = 3

#可选的sysklogd参数
#SYSKLOGD_PARMS =“-m 0”

#控制台参数
#UNICODE = 1
#KEYMAP =“ de-latin1”
#KEYMAP_CORRECTIONS =“ euro2”
#FONT =“ lat0-16 -m 8859-15”
#LEGACY_CHARSET =
```

7.6.8.1。自定义启动和关闭脚本

LFS引导脚本以相当有效的方式引导和关闭系统，但是您可以在rc.site文件中进行一些调整，以进一步提高速度并根据您的喜好调整消息。为此，请在/etc/sysconfig/rc.site上面的文件中调整设置。

- 在启动脚本期间udev，有一个对udev的调用 ，需要一些时间才能完成。根据系统中存在的设备，此时间可能需要也可能不需要。如果您只有简单的分区和单个以太网卡，则引导过程可能不需要等待此命令。要跳过它，请设置变量OMIT_UDEV_SETTLE = y。
- 默认情况下 ，启动脚本udev_retry还会运行udev SETTING。只有在/var单独安装目录的情况下，才默认需要此命令。这是因为时钟需要文件/var/lib/hwclock/adjtime。其他定制也可能需要等待udev完成，但是在许多安装中则不需要。通过设置变量OMIT_UDEV_RETRY_SETTLE = y跳过命令。
- 默认情况下，文件系统检查是静默的。在启动过程中，这似乎是一个延迟。要打开fsck输出，请设置变量VERBOSE_FSCK = y。
- 重新引导时，您可能希望完全跳过文件系统检查fsck。为此，请创建文件/fastboot或使用命令/ sbin / shutdown -f -r now重新启动系统。另一方面，您可以通过使用参数而不是创建/forcefsck或运行 shutdown来强制检查所有文件系统 。 *-F-f*

  设置变量FASTBOOT = y将 在引导过程中禁用 fsck，直到将其删除。不建议永久使用此方法。
- 通常，/tmp启动时会删除目录中的所有文件。根据存在的文件或目录的数量，这可能会导致启动过程中明显的延迟。要跳过删除这些文件的步骤，请设置变量SKIPTMPCLEAN = y。
- 在关闭过程中，初始化程序将TERM信号发送到它已启动的每个程序（例如agetty），等待设置的时间（默认为3秒），然后向每个进程发送KILL信号，然后再次等待。对于未由自己的脚本关闭的任何进程，在sendsignals脚本中重复此过程。初始化延迟可以通过传递参数来设置。例如，要消除init的延迟，请在关闭或重新引导时传递-t0参数（例如 / sbin / shutdown -t0 -r now）。可以通过设置参数KILLDELAY = 0来跳过 sendsignals脚本的延迟。

7.7。Bash Shell启动文件

外壳程序/ bin / bash（以下称为“ 外壳 ”）使用启动文件的集合来帮助创建运行环境。每个文件都有特定的用途，并且可能以不同的方式影响登录和交互环境。目录中的/etc文件提供全局设置。如果主目录中存在等效文件，则它可能会覆盖全局设置。

在成功登录后，使用/ bin / login通过读取/etc/passwd文件来启动交互式登录Shell 。交互式非登录外壳程序从命令行启动（例如[prompt]$/ bin / bash）。运行shell脚本时，通常会出现非交互式shell。它是非交互式的，因为它正在处理脚本并且不等待命令之间的用户输入。

有关更多信息，请参见*Bash启动文件和Interactive Shell*下的info bash。

当外壳程序作为交互式登录外壳程序调用时/etc/profile，~/.bash_profile将读取 文件和 。

/etc/profile下面 的基础设置了一些支持本地语言所必需的环境变量。正确设置它们会导致：

- 程序输出翻译成母语
- 将字符正确分类为字母，数字和其他类别。这是bash在非英语语言环境中的命令行中正确接受非ASCII字符 所必需的
- 正确的国家字母顺序
- 适当的默认纸张尺寸

- 正确设置货币，时间和日期值的格式

替换*<ll>*与所需语言的两个字母的代码（例如，下面的“ 恩 ”），并*<CC>*与相应的国家（例如，双字母代码“ GB ”）。 *<charmap>*应该用您选择的语言环境的规范性charmap代替。 也可以使用可选的修饰符，例如“ @euro ”。

可以通过运行以下命令来获取Glibc支持的所有语言环境的列表：

```
locale -a
```

字符映射可以具有多个别名，例如，“ ISO-8859-1 ”也称为“ iso8859-1 ”和“ iso88591 ”。某些应用程序无法正确处理各种同义词（例如，要求将“ UTF-8 ”写为“ UTF-8 ”，而不是“ utf8 ”），因此在大多数情况下，为特定语言环境选择规范名称是最安全的。要确定规范名称，请运行以下命令，其中*<locale name>*是语言环境-a为您首选的语言环境提供的输出（在我们的示例中为“ en_GB.iso88591 ”）。

```
LC_ALL=<locale name> locale charmap
```

对于“ en_GB.iso88591 ”区域设置，将显示以上命令：

```
ISO-8859-1
```

这将导致最终的语言环境设置为“ en_GB.ISO-8859-1 ”。重要的是，在将使用上述启发式方法发现的语言环境添加到Bash启动文件之前，必须对其进行测试：

```
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
```

上面的命令应该打印语言名称，语言环境使用的字符编码，本地货币以及在进入电话号码之前要拨打的前缀。如果以上任何命令失败并显示与以下所示类似的消息，则表明您的语言环境未在第6章中安装，或默认安装的Glibc不支持。

```
locale: Cannot set LC_* to default locale: No such file or directory
```

如果发生这种情况，您应该使用localedef命令安装所需的语言环境，或考虑选择其他语言环境。进一步的说明假定没有来自Glibc的此类错误消息。

LFS以外的某些软件包也可能不支持您选择的语言环境。一个示例是X库（X窗口系统的一部分），如果语言环境与其内部文件中的字符映射表名称之一不完全匹配，则该库将输出以下错误消息：

```
Warning: locale not supported by Xlib, locale set to C
```

在某些情况下，Xlib希望字符映射表会以大写字母加规范破折号列出。例如，“ ISO-8859-1”而不是“ iso88591”。也可以通过删除语言环境规范的charmap部分来找到合适的规范。可以通过在两个语言环境中运行locale charmap命令来检查。例如，为了使Xlib能够识别此语言环境，必须将“ de_DE.ISO-8859-15@euro”更改为“ de_DE @ euro”。

如果语言环境名称不符合他们的期望，其他软件包也可能无法正常工作（但不一定显示任何错误消息）。在这种情况下，研究其他Linux发行版如何支持您的语言环境可能会提供一些有用的信息。

确定正确的语言环境设置后，创建/etc/profile文件：

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

该“ Ç ”（默认）和“ EN_US ”（推荐一个美国英语用户）的语言环境是不同的。“ C ”使用US-ASCII 7位字符集，并将设置了高位的字节视为无效字符。这就是为什么，例如，ls命令在该语言环境中用问号替换它们。此外，尝试从Mutt或Pine发送带有此类字符的邮件会导致发送不符合RFC的消息（外发邮件中的字符集表示为“ 未知的8位 ”）。因此， 只有在确定不再需要8位字符时，才可以使用“ C ”语言环境。

一些程序不能很好地支持基于UTF-8的语言环境。目前正在努力记录文档，并在可能的情况下解决此类问题，请参阅 **http://www.linuxfromscratch.org/blfs/view/9.0/introduction/locale-issues.html**。

7.8。创建/ etc / inputrc文件

该inputrc文件是Readline库的配置文件，当用户从终端输入行时，该文件提供编辑功能。它通过将键盘输入转换为特定操作来工作。Bash和大多数其他Shell以及许多其他应用程序都使用Readline。

大多数人不需要特定于用户的功能，因此下面的命令将创建一个全局/etc/inputrc名称，供登录的每个人使用。如果以后您决定需要基于每个用户覆盖默认设置，则可以.inputrc在用户的主目录中创建文件修改后的映射。

有关如何编辑inputrc文件的更多信息，请参见*Readline Init File* 部分下的 info bash。信息阅读热线也是很好的信息来源。

以下是一个通用的全局inputrc 注释以及解释各种选项的注释。请注意，注释不能与命令放在同一行。使用以下命令创建文件：

```
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。创建/ etc / shells文件

该shells文件包含系统上的登录Shell列表。应用程序使用此文件来确定外壳程序是否有效。对于每个外壳程序，应该存在一行，其中包含外壳程序相对于目录结构（/）根的路径。

例如，chsh会查询该文件，以确定无特权的用户是否可以更改其自己帐户的登录shell。如果未列出命令名称，则将拒绝用户更改。

对于诸如GDM之类的应用程序（如果找不到则不填充人脸浏览器）/etc/shells或FTP守护程序（传统上不允许访问该文件中未包含的shell的用户）的要求是必需的。

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

第8章使LFS系统可引导

8.1。介绍

现在该使LFS系统可启动了。本章讨论如何创建fstab文件，为新的LFS系统构建内核以及安装GRUB引导加载程序，以便可以选择LFS系统以在启动时进行引导。

8.2。创建/ etc / fstab文件

/etc/fstab某些程序使用 该文件来确定默认情况下文件系统的安装位置，安装顺序以及安装前必须检查的文件系统（完整性错误）。创建一个新的文件系统表，如下所示：

```
cat > /etc/fstab << "EOF"
# Begin /etc/fstab

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

/dev/<xxx>     /            <fff>    defaults            1     1
/dev/<yyy>     swap         swap     pri=1               0     0
proc           /proc        proc     nosuid,noexec,nodev 0     0
sysfs          /sys         sysfs    nosuid,noexec,nodev 0     0
devpts         /dev/pts     devpts   gid=5,mode=620      0     0
tmpfs          /run         tmpfs    defaults            0     0
devtmpfs       /dev         devtmpfs mode=0755,nosuid    0     0

# End /etc/fstab
EOF
```

取代*<xxx>*， *<yyy>*和 *<fff>*与所述的值适合于系统，例如sda2，sda5，和 *ext4*。有关此文件中六个字段的详细信息，请参见man 5 fstab。

具有MS-DOS或Windows起源的文件系统（即vfat，ntfs，smbfs，cifs，iso9660，udf）需要特殊选项utf8，以便正确解释文件名中的非ASCII字符。对于非UTF-8语言环境，iocharset应将的值设置为与语言环境的字符集相同，并以内核能够理解的方式进行调整。如果相关的字符集定义（在配置内核时位于文件系统->本机语言支持下）已编译到内核中或作为模块构建，则此方法有效。但是，如果语言环境的字符集为UTF-8，则对应的选项iocharset=utf8会使文件系统区分大小写。要解决此utf8问题iocharset=utf8，请对UTF-8语言环境使用特殊选项而不是。在“代码页”也为vfat和smbfs文件系统所选项。它应该设置为您所在国家的MS-DOS下使用的代码页号。例如，要安装USB闪存驱动器，ru_RU.KOI8-R用户需要在其安装线的options部分中包含以下内容 /etc/fstab：

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

ru_RU.UTF-8用户的相应选项片段为：

```
noauto,user,quiet,showexec,codepage=866,utf8
```

请注意，using iocharset是的默认设置iso8859-1（使文件系统不区分大小写），该utf8选项告知内核使用UTF-8转换文件名，以便可以在UTF-8语言环境中解释它们。

在内核配置期间，还可以为某些文件系统指定默认的代码页和iocharset值。相关参数名为“默认NLS选项”（CONFIG_NLS_DEFAULT），“默认远程NLS选项”（CONFIG_SMB_NLS_DEFAULT），“FAT的默认代码页”（CONFIG_FAT_DEFAULT_CODEPAGE）和“FAT的默认iocharset”（CONFIG_FAT_DEFAULT_IOCHARSET）。无法为内核编译时的ntfs文件系统。

对于某些硬盘类型，有可能使ext3文件系统在断电时可靠。为此，请将barrier=1mount选项添加 到中的相应条目/etc/fstab。要检查磁盘驱动器是否支持此选项，请在适用的磁盘驱动器上运行 **hdparm**。例如，如果：

```
hdparm -I /dev/sda | grep NCQ
```

返回非空输出，该选项受支持。

注意：基于逻辑卷管理（LVM）的分区不能使用该barrier选项。

8.3。Linux-5.2.8

Linux软件包包含Linux内核。

大概	编译
时间：	4.4-66.0 SBU（通常约6 SBU）
所需的磁盘空间：	960-4250 MB（通常约为1100 MB）

8.3.1。内核安装

构建内核涉及几个步骤-配置，编译和安装。阅读README内核源代码树中的文件，以了解本书配置内核方式的替代方法。

通过运行以下命令准备编译：

```
make mrproper
```

这样可以确保内核树绝对干净。内核团队建议在每次编译内核之前发出此命令。解压后不要依赖源树是否干净。

通过菜单驱动的界面配置内核。有关内核配置的一般信息，请参见 **http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt**。BLFS在 **http://www.linuxfromscratch.org/blfs/view/9.0/longindex.html#kernel-config-index上提供了**有关LFS之外的软件包的特定内核配置要求的一些信息 。有关配置和构建内核的其他信息，**请访问http://www.kroah.com/lkn/。**

> **注意**
>
> 设置内核配置的一个很好的起点是运行make defconfig。这会将基本配置设置为良好状态，从而将您当前的系统体系结构考虑在内。
>
> 确保启用/禁用/设置以下功能，否则系统可能无法正常运行或无法启动：

```
设备驱动程序--->
  通用驱动程序选项--->
   []支持uevent帮助器[CONFIG_UEVENT_HELPER]
   [*]维护一个devtmpfs文件系统以挂载在/ dev [CONFIG_DEVTMPFS]

内核黑客--->
       选择内核展开器（框架指针展开器）---> [CONFIG_UNWINDER_FRAME_POINTER]
```

根据系统要求，可能还需要其他几个选项。有关BLFS软件包所需的选项列表，请参阅 **BLFS内核设置索引**（http://www.linuxfromscratch.org/blfs/view/9.0/longindex.html#kernel-config-index）。

> **注意**
>
> 如果您的主机硬件使用的是UEFI，则上面的'make defconfig'应该会自动添加一些与EFI相关的内核选项。
>
> 为了允许从主机的UEFI引导环境中引导LFS内核，您的内核必须选择以下选项：

```
处理器类型和功能--->
   [*] EFI存根支持[CONFIG_EFI_STUB]
```

lfs-uefi.txt提示（位于 **http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt**）涵盖了从LFS内部管理UEFI环境的完整说明。

以上配置项的基本原理：

Support for uevent helper

使用Udev / Eudev时，设置此选项可能会干扰设备管理。

Maintain a devtmpfs

这将创建由内核填充的自动化设备节点，即使没有运行Udev也是如此。然后，Udev在此之上运行，管理权限并添加符号链接。Udev / Eudev的所有用户都需要此配置项。

```
make menuconfig
```

可选的make环境变量的含义：

LANG=<host_LANG_value> LC_ALL=

这会将语言环境设置建立为主机上使用的语言环境设置。在UTF-8 linux文本控制台上正确的menuconfig ncurses界面线条图可能需要此功能。如果使用，请确保替换*<host_LANG_value>*为$LANG 主机中变量的值。您也可以改用主机的值$LC_ALL或$LC_CTYPE。

另外，在某些情况下，make oldconfig可能更合适。有关README更多信息，请参见文件。

如果需要，可以通过将内核配置文件.config从主机系统（假设它可用）复制到解压缩linux-5.2.8目录中来 跳过内核配置。但是，我们不建议使用此选项。通常，最好浏览所有配置菜单并从头开始创建内核配置。

编译内核映像和模块：

```
make
```

如果使用内核模块，则/etc/modprobe.d可能需要配置模块。与模块和内核配置有关的信息位于第7.3节"设备和模块处理概述"以及该linux-5.2.8/Documentation目录中的内核文档中。同样， modprobe.d(5)可能会引起您的兴趣。

如果内核配置使用了模块，请安装这些模块：

```
make modules_install
```

内核编译完成后，需要其他步骤才能完成安装。有些文件需要复制到/boot目录中。

警告

如果主机系统具有单独的/ boot分区，则下面复制的文件应放在该目录中。最简单的方法是在继续操作之前，将主机（在chroot外部）上的/ boot绑定到/ mnt / lfs / boot。作为***主机系统中***的root用户 ：

```
mount --bind /boot /mnt/lfs/boot
```

内核映像的路径可能会因所使用的平台而异。可以根据您的喜好更改以下文件名，但文件名的主干应为 *vmlinuz*，以与下一节中介绍的引导过程的自动设置兼容。以下命令假定使用x86体系结构：

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-5.2.8-lfs-9.0
```

System.map是内核的符号文件。它映射内核API中每个函数的函数入口点，以及正在运行的内核的内核数据结构的地址。在调查内核问题时，它用作资源。发出以下命令来安装映射文件：

```
cp -iv System.map /boot/System.map-5.2.8
```

上面.config的make menuconfig步骤生成 的内核配置文件包含刚刚编译的内核的所有配置选择。保留此文件以供将来参考是一个好主意：

```
cp -iv .config /boot/config-5.2.8
```

安装Linux内核的文档：

```
install -d /usr/share/doc/linux-5.2.8
cp -r Documentation/* /usr/share/doc/linux-5.2.8
```

重要的是要注意内核源目录中的文件不是*root*拥有的。每当以*root*用户身份打开软件包时，（就像我们在chroot中所做的一样），这些文件具有打包程序计算机上的用户ID和组ID。对于要安装的任何其他软件包，这通常不是问题，因为在安装后会删除源树。但是，Linux源代码树通常会保留很长时间。因此，有可能将打包程序使用的任何用户ID分配给计算机上的某人。该人员随后将拥有对内核源代码的写权限。

注意

在许多情况下，将需要为以后将在BLFS中安装的软件包更新内核的配置。与其他软件包不同，在安装了新构建的内核之后，无需删除内核源代码树。

如果要保留内核源代码树 ，请在目录上运行 chown -R 0：0linux-5.2.8以确保所有文件均由*root*用户拥有。

警告

一些内核文档建议通过/usr/src/linux指向内核源目录来创建符号链接。这是针对2.6系列之前的内核的，并且***不能***在LFS系统上创建，因为它会给基础LFS系统完成后可能希望构建的软件包带来问题。

警告

系统include目录（ /usr/include ）中的标头应***始终***是针对Glibc进行编译的标头，即，在第6.7节“ Linux-5.2.8 API标头”中安装的经过清理的标头。因此，***决不***能将它们替换为原始内核头文件或任何其他经过内核清理的头文件。

8.3.2。配置Linux模块加载顺序

大多数时候，Linux模块是自动加载的，但是有时它需要一些特定的方向。加载模块，程序modprobe的或insmod的，使用/etc/modprobe.d/usb.conf用于这一目的。需要创建此文件，以便如果USB驱动程序（ehci_hcd，ohci_hcd和uhci_hcd）已作为模块构建，则它们将以正确的顺序加载；ehci_hcd必须在ohci_hcd和uhci_hcd之前加载，以避免在引导时输出警告。

/etc/modprobe.d/usb.conf通过运行以下 命令创建一个新文件：

```
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-5.2.8，vmlinuz-5.2.8-lfs-9.0和System.map-5.2.8

安装目录： /lib/modules、/usr/share/doc/linux-5.2.8

简短说明

config-5.2.8	包含内核的所有配置选择
vmlinuz-5.2.8-lfs-9.0	Linux系统的引擎。打开计算机电源时，内核是操作系统的第一部分。它检测并初始化计算机硬件的所有组件，然后将这些组件以文件树的形式提供给软件，并将单个CPU转变为能够同时运行多个程序的多任务计算机。
System.map-5.2.8	地址和符号列表；它映射了内核中所有函数和数据结构的入口点和地址

8.4。使用GRUB设置引导过程

8.4.1。介绍

警告

错误配置GRUB会使您的系统在没有备用引导设备（例如CD-ROM）的情况下无法运行。引导LFS系统不需要此部分。您可能只想修改当前的引导加载程序，例如Grub-Legacy，GRUB2或LILO。

如果计算机无法使用（无法启动），请 确保准备了紧急启动磁盘以“ 抢救 ”计算机。如果您还没有启动设备，则可以创建一个。为了使以下过程正常运行，您需要跳至BLFS并xorriso从 **libisoburn**软件包进行安装。

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

注意

要在启用UEFI的主机系统上引导LFS，需要使用上一节中所述的CONFIG_EFI_STUB功能构建内核。但是，无需使用GRUB2即可引导LFS。为此，需要关闭主机系统BIOS中的UEFI模式和安全启动功能。有关详细信息，请参见 http://www.linuxfromscratch.org/hints/downloads/files/lfs-uefi.txt上 **的lfs-uefi.txt提示**。

8.4.2。GRUB命名约定

GRUB以 *（hdn，m）*的形式为驱动器和分区使用其自己的命名结构，其中*n*是硬盘驱动器号， *m*是分区号。硬盘驱动器号从零开始，但是对于普通分区，分区号从一开始，对于扩展分区，分区号从五开始。请注意，这与两个数字都从零开始的早期版本不同。例如，分区sda1是 GRUB的 *（hd0,1）*，而分区sdb3是 *（hd1,3）*。与Linux相比，GRUB不将CD-ROM驱动器视为硬盘驱动器。例如，如果使用CD hdb 并使用第二个硬盘驱动器hdc，则该第二个硬盘驱动器仍为 *（hd1）*。

8.4.3。设置配置

GRUB通过将数据写入硬盘的第一个物理磁道来工作。该区域不属于任何文件系统。那里的程序访问引导分区中的GRUB模块。缺省位置是/ boot / grub /。

引导分区的位置是影响配置的用户的选择。一种建议是有一个单独的小分区（建议大小为100 MB），仅用于引导信息。这样，无论是LFS还是某些商业发行版，每个构建都可以访问相同的引导文件，并且可以从任何引导的系统进行访问。如果选择执行此操作，则需要安装单独的分区，将当前/boot 目录中的所有文件（例如，您在上一节中刚刚构建的linux内核）移动到新分区。然后，您需要卸载分区并将其重新安装为/boot。如果这样做，请确保进行更新 /etc/fstab。

使用当前的lfs分区也可以，但是配置多个系统更加困难。

使用以上信息，确定根分区（或引导分区，如果使用单独的分区）的适当指示符。对于以下示例，假定根（或单独的引导）分区为 sda2。

将GRUB文件安装到/boot/grub并设置引导轨道：

警告

以下命令将覆盖当前的引导加载程序。如果不需要，请不要运行该命令，例如，如果使用第三方启动管理器来管理主启动记录（MBR）。

```
grub-install /dev/sda
```

注意

如果系统是使用UEFI引导的，则grub-install将尝试安装*x86_64-efi*目标的文件，但是在第6章中尚未安装这些文件。如果是这种情况，请添加--target i386-pc到上面的命令中。

8.4.4。创建GRUB配置文件

产生/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 5.2.8-lfs-9.0" {
        linux   /boot/vmlinuz-5.2.8-lfs-9.0 root=/dev/sda2 ro
}
EOF
```

注意

从GRUB的角度来看，内核文件是相对于所使用的分区的。如果您使用了单独的/ boot分区，请从上述*linux*行中删除/ boot 。您还需要更改*设置的根*行以指向启动分区。

GRUB是一个功能非常强大的程序，它提供了许多选项，可以从各种设备，操作系统和分区类型进行引导。还有许多用于自定义的选项，例如图形启动屏幕，播放声音，鼠标输入等。这些选项的详细信息不在本介绍的范围之内。

警告

有一个命令grub-mkconfig，可以自动写入配置文件。它在/etc/grub.d/中使用了一组脚本，并将破坏您进行的所有自定义。这些脚本主要设计用于非源分发，不建议用于LFS。如果您安装商业Linux发行版，则很有可能会运行该程序。确保备份您的grub.cfg文件。

第9章结束

9.1。结束

做得好！新的LFS系统已安装！我们希望您能使用崭新的定制Linux系统取得圆满成功。

创建/etc/lfs-release文件可能是一个好主意。通过拥有此文件，对于您（以及对于我们是否需要寻求帮助）来说，很容易找出系统上安装了哪个LFS版本。通过运行以下命令创建此文件：

```
echo 9.0 > /etc/lfs-release
```

创建一个文件来显示新系统相对于Linux Standards Base（LSB）的状态也是一个好主意。要创建此文件，请运行：

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="9.0"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

确保对字段" DISTRIB_CODENAME"进行某种自定义，以使系统成为您的唯一系统。

9.2。计数

既然您已经读完了本书，您是否想被算作LFS用户？前往**http://www.linuxfromscratch.org/cgi-bin/lfscounter.php** 并通过输入您的姓名和您使用的第一个LFS版本注册为LFS用户。

现在让我们重启进入LFS。

9.3。重新启动系统

现在已经安装了所有软件，是时候重新启动计算机了。但是，您应该注意一些事情。您在本书中创建的系统非常少，并且很可能没有继续前进所需要的功能。通过在仍然处于我们当前chroot环境中的情况下从BLFS手册中安装一些额外的软件包，可以使自己处于更好的位置，以便在重新启动到新的LFS安装后继续操作。这里有一些建议：

- 文本模式浏览器（例如 **Lynx** ）将使您可以在一个虚拟终端中轻松查看BLFS书籍，而在另一个虚拟终端中构建软件包。
- 该 **GPM**包将允许您执行复制/粘贴到你的虚拟终端的操作。
- 如果您处于静态IP配置无法满足网络要求的情况，则安装 **dhcpcd**或 **dhcp**的客户端部分之类的软件包可能会很有用。
- 安装 **sudo**对于以非root用户身份构建软件包并在新系统中轻松安装生成的软件包可能很有用。
- 如果要在舒适的GUI环境中从远程系统访问新系统，请安装 **openssh**。
- 为了使通过Internet提取文件更容易，请安装 **wget**。
- 如果您的一个或多个磁盘驱动器具有GUID分区表（GPT），则 **gptfdisk**或 **parted**都将很有用。
- 最后，此时还适合查看以下配置文件。
  - / etc / bashrc
  - / etc / dircolors
  - / etc / fstab
  - / etc / hosts
  - / etc / inputrc
  - / etc / profile
  - /etc/resolv.conf
  - / etc / vimrc
  - /root/.bash_profile
  - /root/.bashrc
  - /etc/sysconfig/ifconfig.eth0

既然我们已经说过了，那么让我们开始引导新的LFS安装。首先退出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
```

如果创建了多个分区，请先卸载其他分区再卸载主分区，如下所示：

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

现在，使用以下命令重新引导系统：

```
shutdown -r now
```

假设已按照前面概述的步骤设置了GRUB引导加载程序，则该菜单将设置为自动引导*LFS 9.0*。

重新启动完成后，即可使用LFS系统，并且可能会添加更多软件以满足您的需求。

9.4。现在怎么办？

感谢您阅读这本LFS书籍。我们希望您发现这本书对您有所帮助，并且进一步了解了系统创建过程。

现在已经安装了LFS系统，您可能想知道“ 接下来是 什么？”为了回答这个问题，我们为您整理了一份资源清单。

- 保养

  定期报告所有软件的错误和安全通知。由于LFS系统是从源代码编译而来的，因此您需要及时了解此类报告。有许多在线资源可跟踪此类报告，其中一些如下所示：

  - **CERT**（计算机紧急响应小组）

    CERT具有一个邮件列表，该列表发布有关各种操作系统和应用程序的安全警报。订阅信息可从**http://www.us-cert.gov/cas/signup.html获得**。
  - 布格特拉克

    Bugtraq是完整披露的计算机安全邮件列表。它发布新发现的安全问题，并偶尔发布针对它们的潜在修复程序。订阅信息可从**http://www.securityfocus.com/archive获得**。

- 从零开始超越Linux

  Beyond Linux From Scratch本书涵盖了LFS本书范围之外的各种软件的安装过程。BLFS项目位于**http://www.linuxfromscratch.org/blfs/**。

- LFS提示

LFS提示是LFS社区的志愿者提交的教育文件的集合。这些提示可从**http://www.linuxfromscratch.org/hints/list.html获得**。

- 邮件列表

  如果您需要帮助，想了解最新动态，想为项目做贡献等等，可以订阅几个LFS邮件列表。有关更多信息，请参见第1章-邮件列表。

- Linux文档项目

  Linux文档计划（TLDP）的目标是就Linux文档的所有问题进行协作。TLDP具有大量的HOWTO，指南和手册页。它位于**http://www.tldp.org/**。

第四部分 附录

附录A.缩略语和术语

阿比	应用程序二进制接口
阿尔法	从零开始的自动化Linux
API	应用程序接口
ASCII码	美国标准信息交换码
的BIOS	基本输入输出系统
无国界医生	从零开始超越Linux
BSD	伯克利软件发行
chroot	改变根
CMOS	互补金属氧化物半导体
COS	服务等级
中央处理器	中央处理器
CRC	循环冗余校验
CVS	并发版本系统
DHCP服务器	动态主机配置协议
域名解析	域名服务
EGA	增强型图形适配器
精灵	可执行和可链接格式
紧急行动	文件结束
均衡器	方程

ext2	第二个扩展文件系统
ext3	第三扩展文件系统
ext4	第四扩展文件系统
常问问题	经常问的问题
FHS	文件系统层次结构标准
先进先出	先进先出
全称	完全合格的域名
的FTP	文件传输协议
国标	千兆字节
海湾合作委员会	GNU编译器集合
GID	组标识符
格林威治标准时间	格林威治标准时间
的HTML	超文本标记语言
集成开发环境	集成驱动电子
电气工程师学会	电气电子工程师学会
IO	输入输出
知识产权	互联网协议
IPC	进程间通讯
IRC	互联网中继聊天
ISO标准	国际标准化组织
互联网服务提供商	互联网服务提供商
KB	千字节
发光二极管	发光二极管
LFS	Linux从零开始
最低位	Linux标准库
兆字节	兆字节
MBR	主引导记录
MD5	讯息摘要5
网卡	网络接口卡
NLS	母语支持
NNTP	网络新闻传输协议
NPTL	本机POSIX线程库

开源软件	开放式音响系统
PCH	预编译头
聚四氟乙烯	Perl兼容的正则表达式
PID	流程标识符
PTY	伪终端
服务质量	服务质量
内存	随机存取存储器
RPC	远程过程调用
实时时钟	实时时钟
事业部	标准建造单位
上合组织	圣克鲁斯行动
SHA1	安全哈希算法1
TLDP	Linux文档项目
TFTP	普通文件传输协议
TLS	线程本地存储
UID	用户标识
遮罩	用户文件创建掩码
USB	通用串行总线
世界标准时间	世界标准时间
UUID	通用唯一标识符
风投	虚拟控制台
显卡	视频图形阵列
室速	虚拟终端

附录B致谢

我们要感谢以下人员和组织对Linux From Scratch项目的贡献。

- **Gerard Beekmans** < **gerard** AT linuxfromscratch D0T org> – LFS创建者
- **Bruce Dubbs** <bdubbs AT linuxfromscratch D0T org> – LFS执行编辑
- **Jim Gifford** <吉姆AT linuxfromscratch D0T org> – CLFS项目联合负责人
- **Pierre Labastie** <pierre AT linuxfromscratch D0T org> – BLFS编辑器和ALFS负责人

- **DJ Lucas** <dj at linuxfromscratch D0T org> – LFS和BLFS编辑器
- **Ken Moffat** <ken AT linuxfromscratch D0T org> – BLFS编辑器
- 各种各样的LFS和BLFS邮件列表上的其他人，通过给出建议，测试书并提交错误报告，说明以及他们安装各种软件包的经验，使这本书成为可能。

译者

- **Manuel Canales Esparcia** <macana AT macana-es D0T com> –西班牙LFS翻译项目
- **Johan Lenglet** <johan AT linuxfromscratch D0T org> –直到2008年的法语LFS翻译项目
- **Jean-Philippe Mengual** <jmengual AT linuxfromscratch D0T org> –法国LFS翻译项目2008-2016
- **Julien Lepiller** <jlepiller AT linuxfromscratch D0T org> –法国LFS翻译项目2017年至今
- **Anderson Lizardo** <lizardo AT linuxfromscratch D0T org> –葡萄牙语LFS翻译项目
- **Thomas Reitelbach** <tr erdfunkstelle D0T de> –德国LFS翻译项目
- **安东·迈萨克（Anton Maisak）** <info AT linuxfromscratch D0T org D0T ru> –俄语LFS翻译项目
- **Elena Shevcova** < **helen** AT linuxfromscratch D0T org D0T ru> –俄语LFS翻译项目

镜维护者

北美镜

- **斯科特·**科夫顿（**Scott Kveton**） <scott AT osuosl D0T org> – lfs.oregonstate.edu镜像
- **William Astle** <丢失的AT Lw D0T网络> – ca.linuxfromscratch.org镜像
- **Eujon Sellers** <jpolen@rackspace.com> – lfs.introspeed.com镜像
- **贾斯汀·**克涅里姆（**Justin Knierim**） <tim@idge.net> – lfs-matrix.net镜像

南美镜子

- **Manuel Canales Esparcia** <manuel AT linuxfromscratch D0T org> – lfsmirror.lfs-es.info镜像
- **路易斯·猎鹰** <路易斯·猎鹰> – torredehanoi.org镜子

欧洲镜子

- **Guido Passet** <guido AT底漆D0T网络> – nl.linuxfromscratch.org镜像
- **巴斯蒂安·雅克** <baafie AT planet D0T nl> – lfs.pagefault.net镜像

- **斯文·克兰肖夫** <sven D0T cranshoff AT lineo D0T be> – lfs.lineo.be镜像
- 比利时猩红色– lfs.scarlet.be镜子
- **塞巴斯蒂安·福伯恩（Sebastian Faulborn）** <info at Aliensoft D0T org> – lfs.aliensoft.org镜像
- **斯图尔特·福克斯** <stuart AT dontuse D0T ms> – lfs.dontuse.ms镜像
- **Ralf Uhlemann** <admin AT realhost D0T de> – lfs.oss-mirror.org镜像
- **Antonin Sprinzl** <Antonin D0T Sprinzl AT tuwien D0T ac D0T at> – at.linuxfromscratch.org镜像
- **Fredrik Danerklint** <fredan-lfs AT fredan D0T org> – se.linuxfromscratch.org镜像
- **弗兰克** < **弗兰克** AT linuxpourtous D0T com> – lfs.linuxpourtous.com镜像
- **Philippe Baque** <baque AT cict D0T fr> – lfs.cict.fr镜子
- **Vitaly Chekasin** <朝圣者朝圣者D0T ru> – lfs.pilgrims.ru镜像
- **本杰明·海尔（Benjamin Heil）** <konkokt AT wankoo D0T org> – lfs.wankoo.org镜像
- **Anton Maisak** <info AT linuxfromscratch D0T org D0T ru> – linuxfromscratch.org.ru镜像

亚洲镜

- **Satit Phermsawang** <satit AT wbac D0T ac D0T th> – lfs.phayoune.org镜像
- **Shizunet Co.，Ltd。** <info AT shizu-net D0T jp> – lfs.mirror.shizu-net.jp镜像
- **初始化世界** <http://www.initworld.com/> – lfs.initworld.com镜像

澳大利亚镜子

- **Jason Andrade** < **Jason** AT dstc D0T edu D0T au> – au.linuxfromscratch.org镜像

前项目团队成员

- **Christine Barczak** <theladyskye AT linuxfromscratch D0T org> – LFS图书编辑器
- Archaic <archaic@linuxfromscratch.org> – LFS技术作家/编辑，HLFS项目负责人，BLFS编辑器，提示和补丁项目维护者
- **Matthew Burgess** <matthew AT linuxfromscratch D0T org> – LFS项目负责人，LFS技术作家/编辑
- **Nathan Coulson** <nathan AT linuxfromscratch D0T org> – LFS-引导脚本维护者
- 蒂莫西·鲍彻（Timothy Bauscher）

- 罗伯特·布里格斯
- 伊恩·奇尔顿
- **Jeroen Coumans** <jeroen AT linuxfromscratch D0T org> –网站开发人员，常见问题解答维护者
- **Manuel Canales Esparcia** <manuel AT linuxfromscratch D0T org> – LFS / BLFS / HLFS XML和XSL维护者
- Alex Groenewoud – LFS技术作家
- 马克·海尔丁克
- **Jeremy Huntwork** <jhuntwork AT linuxfromscratch D0T org> – LFS技术作家，LFS LiveCD维护者
- **Bryan Kadzban** <bryan AT linuxfromscratch D0T org> – LFS技术作家
- 马克·海默斯
- Seth W. Klein –常见问题解答维护者
- **尼古拉斯·莱普（Nicholas Leippe）** <nicholas AT linuxfromscratch D0T org> –维基维护者
- **Anderson Lizardo** <lizardo AT linuxfromscratch D0T org> –网站后端脚本维护者
- **Randy McMurchy** <randy AT linuxfromscratch D0T org> – BLFS项目负责人，LFS编辑器
- **Dan Nicholson** <dnicholson AT linuxfromscratch D0T org> – LFS和BLFS编辑器
- **Alexander E. Patrakov** <alexander AT linuxfromscratch D0T org> – LFS技术作家，LFS国际化编辑，LFS Live CD维护者
- 西蒙·佩罗（Simon Perreault）
- **Scot Mc** Pherson <scot AT linuxfromscratch D0T org> – LFS NNTP网关维护者
- **道格拉斯·雷诺（Douglas R. Reno）** <renodr AT linuxfromscratch D0T org> – Systemd编辑器
- **Ryan Oliver** <ryan AT linuxfromscratch D0T org> – CLFS项目联合负责人
- **Greg Schafer** <gschafer AT zip D0T com D0T au> – LFS技术作家和下一代64位支持构建方法的架构师
- Jesse Tie-Ten-Quee – LFS技术作家
- **詹姆斯·罗伯逊（James Robertson）** <jwrober AT linuxfromscratch D0T org> – Bugzilla维护者
- **Tushar Teredesai** <tushar AT linuxfromscratch D0T org> – BLFS书籍编辑，提示和补丁项目负责人
- **Jeremy Utley** <jeremy AT linuxfromscratch D0T org> – LFS技术作家，Bugzilla维护者，LFS-Bootscripts维护者
- **Zack Winkles** <zwinkles AT gmail D0T com> – LFS技术作家

附录C.依赖性

LFS中内置的每个软件包都依赖一个或多个其他软件包才能正确构建和安装。有些软件包甚至参与循环依赖关系，即第一个软件包依赖于第二个软件包，而第二个软件包又依赖于第一个软件包。由于存在这些依赖性，因此在LFS中构建软件包的顺序非常重要。此页面的目的是记录LFS中内置的每个软件包的依赖性。

对于我们构建的每个程序包，我们列出了三种（有时是四种）依赖项类型。第一个列出了需要哪些其他软件包才能编译和安装相关软件包。第二个列出了除了第一个列出的软件包之外还需要哪些软件包才能运行测试套件。第三个依赖关系列表是软件包，需要在构建和安装此软件包之前将其安装在其最终位置。在大多数情况下，这是因为这些程序包将硬编码指向脚本中二进制文件的路径。如果不是按一定顺序建造的，这可能导致/ tools / bin / [binary]的路径被放置在安装到最终系统的脚本中。这显然是不可取的。

最后的依赖关系列表是LFS中未解决的可选软件包，但对用户可能有用。这些程序包本身可能具有其他强制性或可选依赖性。对于这些依赖项，建议的做法是在LFS手册完成后安装它们，然后返回并重新构建LFS软件包。在某些情况下，BLFS解决了重新安装问题。

Acl

安装取决于： Attr，Bash，Binutils，Coreutils，GCC，Gettext，Grep，M4，Make，Perl，Sed和Texinfo
测试套件取决于： Automake，Diffutils，Findutils和Libtool
必须先安装： Coreutils，Sed，Tar和Vim
可选依赖项： 无

Attr

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Grep，M4，Make，Perl，Sed和Texinfo
测试套件取决于： Automake，Diffutils，Findutils和Libtool
必须先安装： Acl和Libcap
可选依赖项： 无

自动配置

安装取决于： Bash，Coreutils，Grep，M4，Make，Perl，Sed和Texinfo
测试套件取决于： Automake，Diffutils，Findutils，GCC和Libtool
必须先安装： Automake
可选依赖项： Emacs

自动制作

安装取决于： Autoconf，Bash，Coreutils，Gettext，Grep，M4，Make，Perl，Sed和Texinfo
测试套件取决于： Binutils，Bison，Bzip2，DejaGNU，Diffutils，Expect，Findutils，Flex，GCC，Gettext，Gzip，Libtool和Tar
必须先安装： 无
可选依赖项： 无

重击

安装取决于： Bash，Binutils，Bison，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，Make，Ncurses，Patch，Readline，Sed和Texinfo
测试套件取决于： 阴影
必须先安装： 无
可选依赖项： Xorg

公元前

安装取决于： Bash，Binutils，Bison，Coreutils，GCC，Glibc，Grep，Make，Perl和Readline
测试套件取决于： Gawk
必须先安装： Linux内核
可选依赖项： 无

Binutils

安装取决于： Bash，Binutils，Coreutils，Diffutils，File，Gawk，GCC，Glibc，Grep，Make，Perl，Sed，Texinfo和Zlib
测试套件取决于： DejaGNU和Expect
必须先安装： 无
可选依赖项： 无

野牛

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，M4，Make，Perl和Sed
测试套件取决于： Diffutils，Findutils和Flex
必须先安装： Kbd和Tar
可选依赖项： Doxygen（测试套件）

Bzip2

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Glibc，Make和Patch
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

校验

安装取决于： GCC，Grep，Make，Sed和Texinfo
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

Coreutils

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，GMP，Grep，Make，Patch，Perl，Sed和Texinfo
测试套件取决于： Diffutils，E2fsprogs，Findutils，Shadow和Util-linux
必须先安装： Bash，Diffutils，Eudev，Findutils和Man-DB
可选依赖项： Perl Expect和IO：Tty模块（用于测试套件）

DejaGNU

安装取决于： Bash，Coreutils，Diffutils，GCC，Grep，Make和Sed
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

Diffutils

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Gettext，Glibc，Grep，Make，Sed和Texinfo

测试套件取决于： Perl
必须先安装： 无
可选依赖项： 无

E2fsprogs

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，Gzip，Make，Sed，Texinfo和Util-linux
测试套件取决于： Procps-ng和Psmisc
必须先安装： 无
可选依赖项： 无

尤德夫

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Glibc，Grep，Gperf，Make和Sed
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

外籍人士

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Glibc，Grep，Make和Sed
测试套件取决于： 无
必须先安装： XML :: Parser
可选依赖项： 无

期望

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Glibc，Grep，Make，Patch，Sed和Tcl
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

文件

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，Make，Sed和Zlib
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

Findutils

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： DejaGNU，Diffutils和Expect
必须先安装： 无
可选依赖项： 无

柔性

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，M4，Make，Patch，Sed和Texinfo
测试套件取决于： Bison和Gawk
必须先安装： IPRoute2，Kbd和Man-DB

可选依赖项： 无

高克

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，GMP，Grep，Make，MPFR，Patch，Readline，Sed和Texinfo
测试套件取决于： Diffutils
必须先安装： 无
可选依赖项： 无

海湾合作委员会

安装取决于： Bash，Binutils，Coreutils，Diffutils，Findutils，Gawk，GCC，Gettext，Glibc，GMP，Grep，M4，Make，MPC，MPFR，Patch，Perl，Sed，Tar和Texinfo
测试套件取决于： DejaGNU，Expect和Shadow
必须先安装： 无
可选依赖项： **GNAT**和**ISL**

GDBM

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Grep，Make和Sed
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

文字

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： Diffutils，Perl和Tcl
必须先安装： Automake
可选依赖项： 无

格里布

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Gettext，Grep，Gzip，Linux API标头，Make，Perl，Python，Sed和Texinfo
测试套件取决于： 文件
必须先安装： 无
可选依赖项： 无

GMP

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，M4，Make，Sed和Texinfo
测试套件取决于： 无
必须先安装： MPFR和GCC
可选依赖项： 无

Gperf

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc和Make
测试套件取决于： Diffutils和Expect
必须先安装： 无

可选依赖项： 无

格列普

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Gettext，Glibc，Grep，Make，Patch，Sed和Texinfo
测试套件取决于： Gawk
必须先安装： Man-DB
可选依赖项： Pcre

格罗夫

安装取决于： Bash，Binutils，Bison，Coreutils，Gawk，GCC，Glibc，Grep，Make，Patch，Sed和Texinfo
测试套件取决于： 没有可用的测试套件
必须先安装： Man-DB和Perl
可选依赖项： GPL Ghostscript

格鲁布

安装取决于： Bash，Binutils，Bison，Coreutils，Diffutils，GCC，Gettext，Glibc，Grep，Make，Ncurses，Sed，Texinfo和Xz
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

压缩文件

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： Diffutils和Less
必须先安装： Man-DB
可选依赖项： 无

依阿那

安装取决于： Coreutils，Gawk和Make
测试套件取决于： 没有可用的测试套件
必须先安装： Perl
可选依赖项： 无

Inetutils

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Grep，Make，Ncurses，Patch，Sed，Texinfo和Zlib
测试套件取决于： 没有可用的测试套件
必须先安装： Tar
可选依赖项： 无

国际工具

安装取决于： Bash，Gawk，Glibc，Make，Perl，Sed和XML :: Parser
测试套件取决于： Perl
必须先安装： 无
可选依赖项： 无

IP路由2

安装取决于： Bash，Bison，Coreutils，Flex，GCC，Glibc，Make和Linux API标头
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

千比特

安装取决于： Bash，Binutils，Bison，Check，Coreutils，Flex，GCC，Gettext，Glibc，Gzip，Make，Patch和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

科莫德

安装取决于： Bash，Binutils，Bison，Coreutils，Flex，GCC，Gettext，Glibc，Gzip，Make，Pkg-config，Sed，Xz-Utils和Zlib
测试套件取决于： 没有可用的测试套件
必须先安装： Eudev
可选依赖项： 无

减

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Glibc，Grep，Make，Ncurses和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： Gzip
可选依赖项： Pcre

Libcap

安装取决于： Attr，Bash，Binutils，Coreutils，GCC，Glibc，Perl，Make和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： Linux-PAM

Libelf

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc和Make
测试套件取决于： 没有可用的测试套件
必须先安装： Linux内核
可选依赖项： 无

利比菲

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Make和Sed
测试套件取决于： DejaGnu
必须先安装： Python
可选依赖项： 无

脂质管道

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，Make，Sed和Texinfo

测试套件取决于： 检查
必须先安装： Man-DB
可选依赖项： 无

Libtool

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： Autoconf，Automake和Findutils
必须先安装： 无
可选依赖项： 无

Linux内核

安装取决于： Bash，Bc，Binutils，Coreutils，Diffutils，Findutils，GCC，Glibc，Grep，Gzip，Kmod，Libelf，Make，Ncurses，OpenSSL，Perl和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

M4

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： Diffutils
必须先安装： Autoconf和Bison
可选依赖项： libsigsegv

使

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： Perl和Procps-ng
必须先安装： 无
可选依赖项： 无

文库

安装取决于： Bash，Binutils，Bzip2，Coreutils，Flex，GCC，GDBM，Gettext，Glibc，Grep，Groff，Gzip，Less，Libpipeline，Make，Sed和Xz
测试套件取决于： Util-linux
必须先安装： 无
可选依赖项： 无

手册页

安装取决于： Bash，Coreutils和Make
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

介子

安装取决于： Ninja和Python

测试套件取决于： 没有可用的测试套件
必须先安装： Systemd
可选依赖项： 无

MPC

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，GMP，Make，MPFR，Sed和Texinfo
测试套件取决于： 无
必须先安装： GCC
可选依赖项： 无

MPFR

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，GMP，Make，Sed和Texinfo
测试套件取决于： 无
必须先安装： Gawk和GCC
可选依赖项： 无

Ncurses

安装取决于： Bash，Binutils，Coreutils，Diffutils，Gawk，GCC，Glibc，Grep，Make，Patch和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： Bash，GRUB，Inetutils，Less，Procps-ng，Psmisc，Readline，Texinfo，Util-linux和Vim
可选依赖项： 无

忍者

安装取决于： Binutils，Coreutils，Gcc和Python
测试套件取决于： 无
必须先安装： Meson
可选依赖项： Asciidoc，Doxygen，Emacs和re2c

Openssl

安装取决于： Binutils，Coreutils，Gcc，Make和Perl
测试套件取决于： 无
必须先安装： Linux
可选依赖项： 无

补丁

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Grep，Make和Sed
测试套件取决于： Diffutils
必须先安装： 无
可选依赖项： Ed

佩尔

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，GDBM，Glibc，Grep，Groff，Make，Sed和Zlib
测试套件取决于： Iana-Etc和Procps-ng
必须先安装： Autoconf

可选依赖项： 无

包配置

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Glibc，Grep，Make，Popt和Sed
测试套件取决于： 无
必须先安装： Kmod
可选依赖项： 无

波普

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Glibc，Grep和Make
测试套件取决于： Diffutils和Sed
必须先安装： Pkg-config
可选依赖项： 无

处理

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Make和Ncurses
测试套件取决于： DejaGNU
必须先安装： 无
可选依赖项： 无

伪造

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，Make，Ncurses和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

蟒蛇

安装取决于： Bash，Binutils，Coreutils，GCC，Gdbm，Gettext，Glibc，Grep，Libffi，Make，Ncurses和Sed
测试套件取决于： GDB和Valgrind
必须先安装： 忍者
可选依赖项： Berkeley DB，OpenSSL，SQLite和Tk

Readline

安装取决于： Bash，Binutils，Coreutils，Gawk，GCC，Glibc，Grep，Make，Ncurses，Patch，Sed和Texinfo
测试套件取决于： 没有可用的测试套件
必须先安装： Bash和Gawk
可选依赖项： 无

塞德

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，Make，Sed和Texinfo
测试套件取决于： Diffutils和Gawk
必须先安装： E2fsprogs，File，Libtool和Shadow
可选依赖项： 无

阴影

安装取决于： Bash，Binutils，Coreutils，Diffutils，Findutils，Gawk，GCC，Gettext，Glibc，Grep，Make和Sed
测试套件取决于： 没有可用的测试套件
必须在之前安装： Coreutils
可选依赖项： Acl，Attr，Cracklib和PAM

Sysklogd

安装取决于： Binutils，Coreutils，GCC，Glibc，Make和Patch
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

系统化

安装取决于： Acl，Attr，Bash，Binutils，Coreutils，Diffutils，Expat，Gawk，GCC，Glibc，Gperf，Grep，Intltool，Libcap，Meson，Sed和Util-linux
测试套件取决于： 无
必须先安装： 无
可选依赖项： 很多，请参见 **BLFS systemd页面**

Sysvinit

安装取决于： Binutils，Coreutils，GCC，Glibc，Make和Sed
测试套件取决于： 没有可用的测试套件
必须先安装： 无
可选依赖项： 无

柏油

安装取决于： Acl，Attr，Bash，Binutils，Bison，Coreutils，GCC，Gettext，Glibc，Grep，Inetutils，Make，Sed和Texinfo
测试套件取决于： Autoconf，Diffutils，Findutils，Gawk和Gzip
必须先安装： 无
可选依赖项： 无

Tcl

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Glibc，Grep，Make和Sed
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

Texinfo

安装取决于： Bash，Binutils，Coreutils，GCC，Gettext，Glibc，Grep，Make，Ncurses，Patch和Sed
测试套件取决于： 无
必须先安装： 无
可选依赖项： 无

实用程序

安装取决于： Bash，Binutils，Coreutils，Diffutils，Eudev，Findutils，Gawk，GCC，Gettext，Glibc，Grep，Make，Ncurses，Sed和Zlib
测试套件取决于： 无
必须先安装： 无
可选依赖项： Libcap-ng

Vim

安装取决于： Acl，Attr，Bash，Binutils，Coreutils，Diffutils，GCC，Glibc，Grep，Make，Ncurses和Sed
测试套件取决于： 无
必须先安装： 无
可选依赖项： Xorg，GTK + 2，LessTif，Python，Tcl，Ruby和GPM

XML ::解析器

安装取决于： Bash，Binutils，Coreutils，Expat，GCC，Glibc，Make和Perl
测试套件取决于： Perl
必须先安装： Intltool
可选依赖项： 无

z

安装取决于： Bash，Binutils，Coreutils，Diffutils，GCC，Glibc和Make
测试套件取决于： 无
必须先安装： Eudev，GRUB，Kmod和Man-DB
可选依赖项： 无

Zlib

安装取决于： Bash，Binutils，Coreutils，GCC，Glibc，Make和Sed
测试套件取决于： 无
必须先安装： File，Kmod，Perl和Util-linux
可选依赖项： 无

附录D.引导和sysconfig脚本版本20190524

本附录中的脚本按它们通常所在的目录列出。顺序是/etc/rc.d/init.d，/etc/sysconfig，/etc/sysconfig/network-devices，和 /etc/sysconfig/network-devices/services。在每个部分中，文件按照通常调用的顺序列出。

D.1。/etc/rc.d/init.d/rc

该rc脚本是init调用的第一个脚本，用于启动引导过程。

```
#！/ bin / bash
############################################### ######################
#开始rc
```

```
#
#说明：主运行级别控制脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#：DJ卢卡斯-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
############################################### ######################

。/ lib / lsb / init-functions

print_error_msg ()
{
   log_failure_msg
   #$ i在调用时设置
   MSG =“ FAILURE：\ n \ n您不应该阅读此错误消息。\ n \ n”
   MSG =“ $ {MSG}，这意味着\ n”中发生了不可预见的错误
   MSG =“ $ {MSG} $ {i}，\ n”
   MSG =“ $ {MSG}，其返回值为$ {error_value}。\ n”

   MSG =“ $ {MSG}如果您能够将此错误追溯到其中一个错误，则\ n”
   MSG =“ $ {MSG} $ {DISTRO_MINI}书提供的文件，\ n”
   MSG =“ $ {MSG}，请随时通过$ {DISTRO_CONTACT}通知我们。\ n”
   log_failure_msg“ $ {MSG}”

   log_info_msg“按Enter继续...”
   wait_for_user
}

check_script_status ()
{
   #$ i在调用时设置
   如果[！-f $ {i}]；然后
      log_warning_msg“ $ {i}不是有效的符号链接。”
      SCRIPT_STAT =“ 1”
   科幻

   如果[！-x $ {i}]；然后
      log_warning_msg“ $ {i}不可执行，正在跳过。”
      SCRIPT_STAT =“ 1”
   科幻
}

跑 ()
```

```
{
   如果[-z $ interactive]; 然后
      $ {1} $ {2}
      返回$?
   科幻

   虽然是真的 做
      阅读-p“运行$ {1} $ {2}（是/否/继续）？” -n 1 runit
      回声

      案例$ {runit}在
         c | C）
            交互式=“”
            $ {i} $ {2}
            ret = $ {? }
            打破;
            ;;

         n | N）
            返回0
            ;;

         y | 是的
            $ {i} $ {2}
            ret = $ {? }
            打破
            ;;
      埃萨克
   做完了

   返回$ ret
}

＃读取任何本地设置/替代
[-r /etc/sysconfig/rc.site] &&源/etc/sysconfig/rc.site

DISTRO = $ {DISTRO：-“ Linux从头开始”}
DISTRO_CONTACT = $ {DISTRO_CONTACT：-“ lfs-dev@linuxfromscratch.org（需要注册）”}
DISTRO_MINI = $ {DISTRO_MINI：-“ LFS”}
IPROMPT = $ {IPROMPT：-“否”}

＃这3个信号不会导致脚本退出
陷阱“” INT QUIT TSTP

[“ $ {1}”！=“”] && runlevel = $ {1}
```

```
如果[“ $ {runlevel}” ==“ ”]; 然后
   echo“用法：$ {0} <runlevel>”>&2
   1号出口
科幻

上一个= $ {PREVLEVEL}
[“ $ {上一个}” ==“ ”] &&上一个= N

如果[! -d /etc/rc.d/rc${runlevel}.d]; 然后
   log_info_msg“ /etc/rc.d/rc${runlevel}.d不存在。\ n”
   1号出口
科幻

如果[[$ runlevel“ ==” 6“ -o” $ runlevel“ ==” 0“]; 然后IPROMPT =“ no”; 科幻

＃注意：在$ {LOGLEVEL：-7}中，它是'：''破折号'7'，而不是负7
如果[“ $ runlevel” ==“ S”]; 然后
   [-r / etc / sysconfig / console] &&源/ etc / sysconfig / console
   dmesg -n“ $ {LOGLEVEL：-7}”
科幻

if [“ $ {IPROMPT}” ==“是” -a“ $ {runlevel}” ==“ S”]; 然后
   ＃发行版欢迎字符串的总长度，不包含转义码
   wlen = $ {wlen：-$（echo“欢迎使用$ {DISTRO}” | wc -c）}
   welcome_message = $ {welcome_message：-“欢迎使用$ {INFO} $ {DISTRO} $ {NORMAL}”}

   ＃交互字符串的总长度，不包含转义码
   ilen = $ {ilen：-$（回显“按'I'进入交互式启动” | wc -c）}
   i_message = $ {i_message：-“按'$ {FAILURE} I $ {NORMAL}'进入交互式启动”}


   ＃dcol和icol是消息之前的空格，以使消息居中
   ＃ 在屏幕上。itime是用户按下按键的等待时间
   wcol = $（（（（$ {COLUMNS}-$ {wlen}）/ 2））
   icol = $（（（（$ {COLUMNS}-$ {ilen}）/ 2））
   itime = $ {itime：-“ 3”}

   回声-e“ \ n \ n”
   echo -e“ \\ 033 [$ {wcol} G $ {welcome_message}”
   echo -e“ \\ 033 [$ {icol} G $ {i_message} $ {NORMAL}”
   回声“”
   读-t“ $ {itime}” -n 1交互式2>＆1> / dev / null
科幻

＃小写
[“ $ {interactive}” ==“ I”] && Interactive =“ i”
```

```
[“ $ {interactive}”！=“ i”] && Interactive =“”

＃从运行级别S读取状态文件（如果存在）
[-r / var / run / interactive] &&源/ var / run / interactive

＃尝试停止由上一个运行级别启动的所有服务，
＃在此运行级别中被杀死
如果[“ $ {previous}”！=“ N”]; 然后
   为我在$（ls -v /etc/rc.d/rc${runlevel}.d/K* 2> / dev / null）中
   做
      check_script_status
      如果[“ $ {SCRIPT_STAT}” ==“ 1”]; 然后
         SCRIPT_STAT =“ 0”
         继续
      科幻

      后缀= $ {i＃/ etc / rc.d / rc $ runlevel.d / K [0-9] [0-9]}
      prev_start = / etc / rc.d / rc $ previous.d / S [0-9] [0-9] $后缀
      sysinit_start = / etc / rc.d / rcS.d / S [0-9] [0-9] $后缀

      如果[“ $ {runlevel}”！=“ 0” -a“ $ {runlevel}”！=“ 6”]; 然后
         如果[！-f $ {prev_start} -a！-f $ {sysinit_start}]; 然后
            MSG =“警告：\ n \ n $ {i}不能为“
            执行MSG =“ $ {MSG}，因为它不是“
            MSG =“ $ {MSG}在上一个“
            MSG =“ $ {MSG}运行级别（$ {previous}）。”
            log_warning_msg“ $ MSG”
            继续
         科幻
      科幻

      运行$ {i}停止
      error_value = $ {？}

      如果[“ $ {error_value}”！=“ 0”]; 然后print_error_msg; 科幻
   做完了
科幻

如果[“ $ {previous}” ==“ N”]; 然后导出IN_BOOT = 1; 科幻

如果[“ $ runlevel” ==“ 6” -a -n“ $ {FASTBOOT}”]；然后
   触摸/ fastboot
科幻


＃启动此运行级别中的所有功能
```

```
对于我在$（ls -v /etc/rc.d/rc${runlevel}.d/S* 2> / dev / null）中
做
   如果[“ $ {previous}”！=“ N”]; 然后
      后缀= $ {i#/ etc / rc.d / rc $ runlevel.d / S [0-9] [0-9]}
      stop = / etc / rc.d / rc $ runlevel.d / K [0-9] [0-9] $后缀
      prev_start = / etc / rc.d / rc $ previous.d / S [0-9] [0-9] $后缀

      [-f $ {prev_start} -a! -f $ {stop}] &&继续
   科幻

   check_script_status
      如果[“ $ {SCRIPT_STAT}” ==“ 1”]; 然后
         SCRIPT_STAT =“ 0”
         继续
      科幻

   案例$ {runlevel}在
      0 | 6）
         运行$ {i}停止
         ;;
      *）
         运行$ {i}开始
         ;;
   埃萨克

   error_value = $ {? }

   如果[“ $ {error_value}”！=“ 0”]; 然后print_error_msg; 科幻
做完了

＃将交互式变量存储在运行级别S的开关上，如果没有，则将其删除
if [“ $ {runlevel}” ==“ S” -a“ $ {interactive}” ==“ i”]; 然后
    echo“ interactive = \” i \“”> / var / run / interactive
其他
    rm -f / var / run / interactive 2> / dev / null
科幻

＃仅在初始启动时复制启动日志
如果[“ $ {previous}” ==“ N” -a“ $ {runlevel}”！=“ S”]; 然后
   cat $ BOOTLOG >> /var/log/boot.log

   ＃标记启动结束
   回声“ --------” >> /var/log/boot.log

   ＃删除临时文件
   rm -f $ BOOTLOG 2> / dev / null
```

```
科幻

#结束rc
```

D.2。/ lib / lsb / init-functions

```
#！/ bin / sh
################################################## ######################
#
#开始/ lib / lsb / init-funtions
#
#说明：运行级别控制功能
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#：DJ卢卡斯-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
#注意：使用基于Matthias Benkmann的simpleinit-msb的代码
#http://winterdrache.de/linux/newboot/index.html
#
#文件应位于/ lib / lsb中
#
################################################## ######################

##环境设置
#设置环境的默认值
umask 022
导出PATH =“ / bin：/ usr / bin：/ sbin：/ usr / sbin”

##设置颜色命令，通过echo使用
#请查阅`man console_codes了解更多信息
#在“ ECMA-48设置图形渲染”部分下
#
#警告：从8位字体切换到9位字体时，
#linux控制台将把粗体（1;）重新解释为
#9bit字体的前256个字形。这确实
#不影响帧缓冲控制台

NORMAL =“ \\ 033 [0; 39m”#标准控制台灰色
SUCCESS =“ \\ 033 [1; 32m”#成功是绿色
WARNING =“ \\ 033 [1; 33m”#警告为黄色
FAILURE =“ \\ 033 [1; 31m”#失败是红色的
```

```
INFO =“ \\ 033 [1; 36m”＃信息为浅青色
BRACKET =“ \\ 033 [1; 34m”＃括号为蓝色

＃使用彩色前缀
BMPREFIX =“”
SUCCESS_PREFIX =“ $ {SUCCESS} * $ {NORMAL}”
FAILURE_PREFIX =“ $ {FAILURE} ***** $ {NORMAL}”
WARNING_PREFIX =“ $ {WARNING} *** $ {NORMAL}”
SKIP_PREFIX =“ $ {INFO} S $ {NORMAL}”

SUCCESS_SUFFIX =“ $ {BRACKET} [$ {SUCCESS} OK $ {BRACKET}] $ {NORMAL}”
FAILURE_SUFFIX =“ $ {BRACKET} [$ {FAILURE} FAIL $ {BRACKET}] $ {NORMAL}”
WARNING_SUFFIX =“ $ {BRACKET} [$ {WARNING} WARN $ {BRACKET}] $ {NORMAL}”
SKIP_SUFFIX =“ $ {BRACKET} [$ {INFO} SKIP $ {BRACKET}] $ {NORMAL}”

BOOTLOG = /运行/启动日志
杀手= 3
SCRIPT_STAT =“ 0”

＃设置任何用户指定的环境变量，例如HEADLESS
[-r /etc/sysconfig/rc.site] &&。/etc/sysconfig/rc.site

##屏幕尺寸
＃查找当前屏幕尺寸
如果[-z“ $ {COLUMNS}”]；然后
   栏= $（固定大小）
   COLUMNS = $ {COLUMNS ## *}
科幻

＃使用远程连接（例如串行端口）时，stty size返回0
如果[“ $ {COLUMNS}” =“ 0”]；然后
   栏= 80
科幻

##定位结果消息的测量
COL = $（（（$ {COLUMNS}-8））））
WCOL = $（（（$ {COL}-2））））

##设置光标位置命令，通过echo使用
SET_COL =“ \\ 033 [$ {COL} G”＃在$ COL字符处
SET_WCOL =“ \\ 033 [$ {WCOL} G”＃在$ WCOL字符处
CURS_UP =“ \\ 033 [1A \\ 033 [0G”＃向上一行，在第0个字符处
CURS_ZERO =“ \\ 033 [0G”

################################################ ##############################
＃start_daemon（）＃
```

```
#用法：start_daemon [-f] [-n nicelevel] [-p pidfile]路径名[args ...]#
##
#目的：这会将指定程序作为守护程序运行#
##
#输入：-f ：（强制）运行程序，即使该程序已经在运行。#
#-n nicelevel：指定一个不错的等级。参见“ man nice（1）”。#
#-p pidfile：使用指定的文件确定PID。#
#pathname：指定程序的完整路径
#args：传递给程序的其他参数（路径名）#
##
#返回值（由LSB退出代码定义）：#
#0-程序正在运行或服务正常#
#1-通用错误或未指定的错误#
#2-参数无效或过多
#5-未安装程序#
################################################ ##############################
start_daemon（）
{
    本地部队=“”
    本地nice =“ 0”
    本地pidfile =“”
    本地pidlist =“”
    本地retval =“”

    #处理参数
    虽然真实
    做
        情况为“ $ {1}”

            -F）
                力=“ 1”
                1班
                ;;

            -n）
                nice =“ $ {2}”
                班次2
                ;;

            -p）
                pidfile =“ $ {2}”
                班次2
                ;;

            -*）
                返回2
```

```
            ;;

        *)
            程序=“ $ {1}”
            打破
            ;;
    埃萨克
做完了

#检查有效程序
如果[! -e“ $ {program}”]; 然后返回5; 科幻

# 执行
如果[-z“ $ {force}”]; 然后
    如果[-z“ $ {pidfile}”]; 然后
        #通过发现确定pid
        pidlist =`pidofproc“ $ {1}”`
        retval =“ $ {? }”
    其他
        #PID文件包含所需的PID
        #注意，按照LSB的要求，必须将路径指定给pidofproc，
        #但是，当前实现或标准未使用它。
        pidlist =`pidofproc -p“ $ {pidfile}” “ $ {1}”``
        retval =“ $ {? }”
    科幻

    #仅返回值
    #这是初始化脚本（或发行版的函数）的职责
    #记录消息！
    案例“ $ {retval}”在

        0）
            #程序已经正确运行，这是一个
            #成功启动。
            返回0
            ;;

        1）
            #程序未运行，但存在无效的pid文件
            #删除pid文件并继续
            rm -f“ $ {pidfile}”
            ;;

        3）
            #程序未运行，不存在pidfile
            #在这里什么也不做，让start_deamon继续。
```

```
                ;;

            *)
                #状态值返回的其他值不得解释
                #并作为未指定的错误返回。
                返回1
                ;;
        埃萨克
    科幻

    #开始!
    不错-n“ $ {nice}” “ $ {@}”
}

################################################ ############################
#killproc（） #
#用法：killproc [-p pidfile]路径名[signal]#
##
#目的：向正在运行的进程发送控制信号#
##
#输入：-p pidfile，使用指定的pidfile#
#路径名，指定程序的路径名
#信号，将此信号发送到路径名#
##
#返回值（由LSB退出代码定义）： #
#0-程序（路径名）已停止/已经停止或#
#正在运行的程序已发送指定信号并已停止#
#成功#
#1-通用错误或未指定的错误#
#2-参数无效或过多
#5-未安装程序#
#7-程序未运行，并且提供了信号#
################################################ ############################
killproc（）
{
    本地pidfile
    本地程序
    本地前缀
    本地姓氏
    本地信号=“-TERM”
    本地fallback =“-KILL”
    本地号码
    本地设备清单
    本地撤回
    本地pid
    本地延迟=“ 30”
```

```
本地死角
当地时间

#处理参数
虽然是真的 做
    情况为“ $ {1}”
        -p）
            pidfile =“ $ {2}”
            班次2
            ;;

         *）
             程序=“ $ {1}”
             如果[-n“ $ {2}”]; 然后
                 signal =“ $ {2}”
                 fallback =“”
             其他
                 nosig = 1
             科幻

             #其他参数错误
             如果[-n“ $ {3}”]; 然后
                 返回2
             其他
                 打破
             科幻
             ;;
    埃萨克
做完了

#检查有效程序
如果[! -e“ $ {program}”]; 然后返回5; 科幻

#检查有效信号
check_signal“ $ {信号}”
如果[“ $ {? }” -ne“ 0”]; 然后返回2; 科幻

#获取pid列表
如果[-z“ $ {pidfile}”]; 然后
    #通过发现确定pid
    pidlist =`pidofproc“ $ {1}”`
    retval =“ $ {? }”
其他
    #PID文件包含所需的PID
    #注意，按照LSB的要求，必须将路径指定给pidofproc，
    #但是，当前实现或标准未使用它。
```

```
    pidlist =`pidofproc -p“ $ {pidfile}” “ $ {1}”``
    retval =“ $ {? }”
科幻

#仅返回值
#这是初始化脚本（或发行版的函数）的职责
#记录消息!
案例“ $ {retval}”在

    0）
        #程序运行正常
        #在这里什么也不做，让killproc继续。
        ;;

    1）
        #程序未运行，但存在无效的pid文件
        #删除pid文件。
        rm -f“ $ {pidfile}”

        #仅在没有信号传递的情况下成功。
        如果[-n“ $ {nosig}”]; 然后
            返回0
        其他
            返回7
        科幻
        ;;

    3）
        #程序未运行，不存在pidfile
        #仅在没有信号传递的情况下成功。
        如果[-n“ $ {nosig}”]; 然后
            返回0
        其他
            返回7
        科幻
        ;;

    *）
        #状态值返回的其他值不得解释
        #并作为未指定的错误返回。
        返回1
        ;;
埃萨克

#对退出信号和控制信号执行不同的操作
check_sig_type“ $ {信号}”
```

```
如果[“ $ {? }” -eq “ 0”]; 然后使用＃Signal终止程序

    ＃帐户为空pidlist（pid文件仍然存在并且没有
    ＃信号已发出）
    如果[“ $ {pidlist}”！=“”]; 然后

        ＃杀死pid列表
        为$ {pidlist}中的pid; 做

            kill -0 “ $ {pid}” 2> / dev / null

            如果[“ $ {? }” -ne “ 0”]; 然后
                ＃过程已死，继续下一步并假设一切都很好
                继续
            其他
                杀死“ $ {signal}” “ $ {pid}” 2> / dev / null

                ＃等待最多$ {delay} / 10秒，以等待“ $ {pid}”
                ＃在十分之一秒内终止

                而[[$ {delay}“ -ne” 0“]; 做
                    kill -0 “ $ {pid}” 2> / dev / null || piddead =“ 1”
                    如果[“ $ {piddead}” =“ 1”]; 然后休息 科幻
                    睡0.1
                    delay =“ $ （（$ $ {delay}-1））”
                做完了

                ＃如果设置了后备功能，并且程序仍在运行，则
                ＃使用后备
                如果[-n “ $ {fallback}” -a “ $ {piddead}”！=“ 1”]; 然后
                    杀死“ $ {fallback}” “ $ {pid}” 2> / dev / null
                    睡觉1
                    ＃再次检查，如果仍在运行则失败
                    kill -0 “ $ {pid}” 2> / dev / null &&返回1
                科幻
            科幻
        做完了
    科幻

    ＃检查并删除过时的PID文件。
    如果[-z “ $ {pidfile}”]; 然后
        ＃查找$ program的基本名称
        prefix =`echo “ $ {program}” | sed's / [^ /] * $ //'`
        progname =`echo “ $ {program}” | sed “ s @ $ {prefix} @@”`
```

```
            如果[-e“ /var/run/${progname}.pid”]；然后
                rm -f“ /var/run/${progname}.pid” 2> / dev / null
            科幻
        其他
            如果[-e“ $ {pidfile}”]；然后rm -f“ $ {pidfile}” 2> / dev / null；科幻
        科幻

    #对于不希望程序退出的信号，只需
    #让kill发挥作用，并评估kill的回报价值

    else#check_sig_type-信号不用于终止程序
        为$ {pidlist}中的pid；做
            杀死“ $ {signal}”“ $ {pid}”
            如果[“ $ {? }” -ne“ 0”]；然后返回1；科幻
        做完了
    科幻
}

################################################ ##############################
#pidofproc（）#
#用法：pidofproc [-p pidfile]路径名#
##
#用途：此函数为特定守护程序返回一个或多个pid#
##
#输入：-p pidfile，使用指定的pidfile代替pidof#
#路径名，指定程序的路径#
##
#返回值（由LSB状态码定义）：#
#0-成功（PID到标准输出）#
#1-程序已死，PID文件仍然存在（剩余PID输出）
#3-程序未运行（无输出）#
################################################ ##############################
pidofproc（）
{
    本地pidfile
    本地程序
    本地前缀
    本地姓氏
    本地设备清单
    本地脂质
    本地exitstatus =“ 0”

    #处理参数
    虽然是真的 做
        情况为“ $ {1}”
```

```
        -p）
            pidfile =“ $ {2}”
            班次2
            ;;

        *）
            程序=“ $ {1}”
            如果[-n“ $ {2}”]；然后
                # 争论太多
                #因为这是状态，所以返回未知
                返回4
            其他
                打破
            科幻
            ;;
    埃萨克
做完了

#如果未指定PID文件，请尝试查找一个。
如果[-z“ $ {pidfile}”]；然后
    #获取程序的基本名称
    prefix =`echo“ $ {program}” | sed's / [^ /] * $ //'``

    如果[-z“ $ {prefix}”]；然后
       progname =“ $ {program}”
    其他
       progname =`echo“ $ {program}” | sed“ s @ $ {prefix} @@”`
    科幻

    #如果存在具有该名称的PID文件，则假定为该名称。
    如果[-e“ /var/run/${progname}.pid”]；然后
        pidfile =“ / var / run / $ {progname} .pid”
    科幻
科幻

#如果PID文件已设置并存在，请使用它。
如果[-n“ $ {pidfile}” -a -e“ $ {pidfile}”]；然后

    #使用pidfile第一行中的值
    pidlist =`/ bin / head -n1“ $ {pidfile}”``
    #可以选择将其写为'sed 1q'代替'head -n1'
    #LFS应该将/ bin / head移到/ usr / bin / head
其他
    #使用pidof
    pidlist =`pidof“ $ {program}”``
科幻
```

```
    #找出所有列出的PID是否正在运行。
    为$ {pidlist}中的pid；做
        kill -0 $ {pid} 2> / dev / null

        如果[“ $ {? }” -eq“ 0”]；然后
            lpids =“ $ {lpids} $ {pid}”
        其他
            exitstatus =“ 1”
        科幻
    做完了

    如果[-z“ $ {lpids}” -a! -f“ $ {pidfile}”]；然后
        返回3
    其他
        回声“ $ {lpids}”
        返回“ $ {exitstatus}”
    科幻
}

################################################ ##############################
#statusproc（）#
#用法：statusproc [-p pidfile]路径名#
##
#目的：此函数将特定守护程序的状态打印到stdout#
##
#输入：-p pidfile，使用指定的pidfile代替pidof#
#路径名，指定程序的路径#
##
#返回值：#
#0-状态打印
#1-输入错误。未指定要检查的守护程序。#
################################################ ##############################
statusproc（）
{
   本地pidfile
   本地设备清单

   如果[“ $ {#}” =“ 0”]；然后
      回声“用法：statusproc [-p pidfle] {程序}”
      1号出口
   科幻

   #处理参数
   虽然是真的 做
       情况为“ $ {1}”
```

```
        -p)
            pidfile =“ $ {2}”
            班次2
            ;;

        *)
            如果[-n“ $ {2}”]; 然后
                回显“参数过多”
                返回1
            其他
                打破
            科幻
            ;;
    埃萨克
做完了

如果[-n“ $ {pidfile}”]; 然后
   pidlist =`pidofproc -p“ $ {pidfile}” $ @`
其他
   pidlist =`pidofproc $ @`
科幻

＃修剪尾随空格
pidlist =`echo“ $ {pidlist}” | sed -r's / + $ //'`

base =“ $ {1 ## * /}”

如果[-n“ $ {pidlist}”]; 然后
   / bin / echo -e“ $ {INFO} $ {base}与Process一起运行” \
       “ ID $ {pidlist}。$ {NORMAL}”
其他
   如果[-n“ $ {base}” -a -e“ /var/run/${base}.pid”]; 然后
      / bin / echo -e“ $ {警告} $ {1}没有运行，但是”
          “ /var/run/${base}.pid存在。$ {NORMAL}”
      其他
         如果[-n“ $ {pidfile}” -a -e“ $ {pidfile}”]; 然后
            / bin / echo -e“ $ {警告} $ {1}没有运行” \
                “但是存在$ {pidfile}。$ {NORMAL}”
         其他
            / bin / echo -e“ $ {INFO} $ {1}未运行。$ {NORMAL}”
         科幻
      科幻
   科幻
}
```

```
################################################ ############################
＃timespec（）＃
＃＃
＃目的：内部实用程序函数，用于格式化时间戳记＃
＃引导日志文件。设置STAMP变量。＃
＃＃
＃返回值：未使用＃
################################################ ############################
timespec（）
{
   STAMP =“ $（echo`date +” %b%d%T%：z“``hostname`）”
   返回0
}

################################################ ############################
＃log_success_msg（）＃
＃用法：log_success_msg [“ message”]＃
＃＃
＃目的：在屏幕上打印成功的状态消息，然后＃
＃引导日志文件。＃
＃＃
＃输入：$ @-消息＃
＃＃
＃返回值：未使用＃
################################################ ############################
log_success_msg（）
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”
    / bin / echo -e“ $ {CURS_ZERO} $ {SUCCESS_PREFIX} $ {SET_COL} $ {SUCCESS_SUFFIX}”

    ＃从日志文件中删除不可打印的字符
    logmessage =`echo“ $ {@}” | sed's / \\\ 033 [^ a-zA-Z] *。// g'`

    时间规格
    / bin / echo -e“ $ {STAMP} $ {logmessage} OK” >> $ {BOOTLOG}

    返回0
}

log_success_msg2（）
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”
    / bin / echo -e“ $ {CURS_ZERO} $ {SUCCESS_PREFIX} $ {SET_COL} $ {SUCCESS_SUFFIX}”

    回声“确定” >> $ {BOOTLOG}
```

```
    返回0
}

################################################ ##############################
＃log_failure_msg（）＃
＃用法：log_failure_msg [“ message”]＃
＃＃
＃目的：在屏幕上打印故障状态消息，然后＃
＃引导日志文件。＃
＃＃
＃输入：$ @-消息＃
＃＃
＃返回值：未使用＃
################################################ ##############################
log_failure_msg（）
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”
    / bin / echo -e“ $ {CURS_ZERO} $ {FAILURE_PREFIX} $ {SET_COL} $ {FAILURE_SUFFIX}”

    ＃从日志文件中删除不可打印的字符

    时间规格
    logmessage =`echo“ $ {@}” | sed's / \\\ 033 [^ a-zA-Z] *。// g'`
    / bin / echo -e“ $ {STAMP} $ {logmessage}失败” >> $ {BOOTLOG}

    返回0
}

log_failure_msg2（）
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”
    / bin / echo -e“ $ {CURS_ZERO} $ {FAILURE_PREFIX} $ {SET_COL} $ {FAILURE_SUFFIX}”

    回声“失败” >> $ {BOOTLOG}

    返回0
}

################################################ ##############################
＃log_warning_msg（）＃
＃用法：log_warning_msg [“ message”]＃
＃＃
＃目的：在屏幕上显示警告状态消息，然后＃
＃引导日志文件。＃
＃＃
＃返回值：未使用＃
```

```
################################################ ############################
log_warning_msg ()
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”
    / bin / echo -e“ $ {CURS_ZERO} $ {WARNING_PREFIX} $ {SET_COL} $ {WARNING_SUFFIX}”

    #从日志文件中删除不可打印的字符
    logmessage =`echo“ $ {@}” | sed's / \\\ 033 [^ a-zA-Z] *。// g'`
    时间规格
    / bin / echo -e“ $ {STAMP} $ {logmessage} WARN” >> $ {BOOTLOG}

    返回0
}

log_skip_msg ()
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”
    / bin / echo -e“ $ {CURS_ZERO} $ {SKIP_PREFIX} $ {SET_COL} $ {SKIP_SUFFIX}”

    #从日志文件中删除不可打印的字符
    logmessage =`echo“ $ {@}” | sed's / \\\ 033 [^ a-zA-Z] *。// g'`
    / bin / echo“跳过” >> $ {BOOTLOG}

    返回0
}

################################################ ############################
#log_info_msg () #
#用法：log_info_msg消息#
##
#目的：在屏幕上打印一条信息消息，然后#
#引导日志文件。不打印尾随换行符。#
##
#返回值：未使用#
################################################ ############################
log_info_msg ()
{
    / bin / echo -n -e“ $ {BMPREFIX} $ {@}”

    #从日志文件中删除不可打印的字符
    logmessage =`echo“ $ {@}” | sed's / \\\ 033 [^ a-zA-Z] *。// g'`
    时间规格
    / bin / echo -n -e“ $ {STAMP} $ {logmessage}” >> $ {BOOTLOG}

    返回0
}
```

```
log_info_msg2 ()
{
    / bin / echo -n -e “ $ {@}”

    ＃从日志文件中删除不可打印的字符
    logmessage =`echo “ $ {@}” | sed's / \\\ 033 [^ a-zA-Z] *。// g'`
    / bin / echo -n -e “ $ {logmessage}” >> $ {BOOTLOG}

    返回0
}

################################################ ##############################
#valuate_retval () ＃
＃用法：评估返回值并适当地打印成功或失败＃
＃＃
＃目的：方便功能，用于终止信息消息＃
＃＃
＃返回值：未使用＃
################################################ ##############################
Evaluation_retval ()
{
   本地error_value =“ $ {? }”

   如果[$ {error_value} = 0]; 然后
      log_success_msg2
   其他
      log_failure_msg2
   科幻
}

################################################ ##############################
＃check_signal () ＃
＃用法：check_signal [-{signal} | {signal}]＃
＃＃
＃用途：检查有效信号。没有任何LSB草案对此进行定义，＃
＃，但需要检查信号以确定＃
选择的＃个信号是其他函数的无效参数。＃
＃＃
＃输入：接受格式为-{signal}或{signal}的单个字符串值
＃＃
＃返回值：＃
＃0-成功（信号有效＃
＃1-信号无效＃
################################################ ##############################
check_signal ()
```

```
{
    当地瓦尔西格

    #添加对无效信号的错误处理
    valsig =“-ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2”
    valsig =“ $ {valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN”
    valsig =“ $ {valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP”
    valsig =“ $ {valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9”
    valsig =“ $ {valsig} -11 -13 -14 -15”

    回声“ $ {valsig}” | grep-“ $ {1}”> / dev / null

    如果[“ $ {? }” -eq“ 0”]; 然后
        返回0
    其他
        返回1
    科幻
}

################################################ ##############################
#check_sig_type（）#
#用法：check_signal [-{signal} | {signal}]#
##
#目的：检查信号是程序终止信号还是控制信号#
#这不是任何LSB草案定义的，但是，要求#
#检查信号以确定它们是否打算结束#
#编程或简单地控制它。#
##
#输入：接受格式为-{signal}或{signal}的单个字符串值
##
#返回值：#
#0-信号用于程序终止#
#1-信号用于程序控制#
################################################ ##############################
check_sig_type（）
{
    当地瓦尔西格

    #终止信号列表（仅限于常用项目）
    valsig =“-ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15”

    回声“ $ {valsig}” | grep-“ $ {1}”> / dev / null

    如果[“ $ {? }” -eq“ 0”]; 然后
        返回0
    其他
```

```
        返回1
    科幻
}

################################################## ##############################
#wait_for_user（）#
##
#目的：如果不是无头系统，请等待用户响应#
##
################################################## ##############################
wait_for_user（）
{
   #默认等待用户
   [“ $ {HEADLESS = 0}” =“ 0”] &&读ENTER
   返回0
}

################################################## ##############################
# 是真的（） #
##
#目的：用来测试变量是否为true的实用工具| 是的 1号
##
################################################## ##############################
是真的（）
{
   [“ $ 1” =“ 1”] || [“ $ 1” =“是”] || [“ $ 1” =“ true”] || [“ $ 1” =“ y”] ||
   [“ $ 1” =“ t”]
}

#结束/ lib / lsb / init-functions
```

D.3。/etc/rc.d/init.d/mountvirtfs

```
#！/ bin / sh
################################################## ######################
#开始mountvirtfs
#
#描述：挂载proc，sysfs并运行
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
```

```
#
############################################### ####################

###开始初始化信息
#提供：mountvirtfs
#必选-开始：
#应该开始：
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：
#简短描述：挂载/ sys和/ proc虚拟（内核）文件系统。
#挂载/ run（tmpfs）和/ dev（devtmpfs）。
#说明：挂载/ sys和/ proc虚拟（内核）文件系统。
#挂载/ run（tmpfs）和/ dev（devtmpfs）。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      #在记录任何消息之前确保/ run可用
      如果！mountpoint / run> / dev / null; 然后
         挂载/运行|| 失败= 1
      科幻

      mkdir -p /运行/锁定/运行/ shm
      chmod 1777 / run / shm / run / lock

      log_info_msg“安装虚拟文件系统：$ {INFO} / run”

      如果！mountpoint / proc> / dev / null; 然后
         log_info_msg2“ $ {INFO} / proc”
         挂载-o nosuid，noexec，nodev / proc || 失败= 1
      科幻

      如果！mountpoint / sys> / dev / null; 然后
         log_info_msg2“ $ {INFO} / sys”
         挂载-o nosuid，noexec，nodev / sys || 失败= 1
      科幻

      如果！mountpoint / dev> / dev / null; 然后
         log_info_msg2“ $ {INFO} / dev”
         挂载-o模式= 0755，nosuid / dev || 失败= 1
      科幻
```

```
      ln -sfn / run / shm / dev / shm

      （退出$ {failed}）
      valuate_retval
      退出$ failed
      ;;

   *）
      回声“用法：$ {0} {开始}”
      1号出口
      ;;
埃萨克

#结束mountvirtfs
```

D.4。/etc/rc.d/init.d/modules

```
#！/ bin / sh
################################################## ######################
#开始模块
#
#说明：模块自动加载脚本
#
#作者：Zack Winkles
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
################################################## ######################

###开始初始化信息
#提供：模块
#必需-开始：mountvirtfs sysctl
#应该开始：
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：
#简短描述：加载所需的模块。
#说明：加载/ etc / sysconfig / modules中列出的模块。
#X-LFS提供-通过：LFS
###结束初始化信息
```

```
#确保内核具有模块支持。
[-e / proc / modules] || 出口0

。/ lib / lsb / init-functions

情况为“ $ {1}”
  开始）
    #如果没有模块文件或没有模块文件，则退出
    #个有效条目
    [-r / etc / sysconfig / modules] || 出口0
    egrep -qv'^（$ |#）'/ etc / sysconfig / modules || 出口0

    log_info_msg“正在加载模块：”

    #仅在用户实际给我们的情况下尝试加载模块
    #加载一些模块。

    在读取模块args时；做

      #忽略注释和空白行。
      案例“ $模块”在
        “” |“#” *）继续;;
      埃萨克

      #尝试通过提供的所有参数加载模块。
      modprobe $ {module} $ {args}> / dev / null

      #如果成功，则打印模块名称，否则请注意。
      如果[$？-eq 0]；然后
        log_info_msg2“ $ {模块}”
      其他
        failedmod =“ $ {failedmod} $ {module}”
      科幻
    完成</ etc / sysconfig / modules

    #在正确的行上显示有关成功加载模块的消息。
    log_success_msg2

    #打印一条失败消息，其中包含所有
    #可能加载失败。
    如果[-n“ $ {failedmod}”]；然后
      log_failure_msg“无法加载模块：$ {failedmod}”
      1号出口
    科幻
    ;;
```

```
   *)
      回声“用法：$ {0} {开始}”
      1号出口
      ;;
埃萨克

出口0

＃终端模块
```

D.5。/etc/rc.d/init.d/udev

```
＃！/ bin / sh
################################################ ######################
＃开始udev
＃
＃说明：Udev冷插脚本
＃
＃作者：Zack Winkles，Alexander E. Patrakov
＃DJ Lucas-dj AT linuxfromscratch DOT org
＃更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
＃
＃版本：LFS 7.0
＃
################################################ ######################

###开始初始化信息
＃提供：udev $ time
＃必选-开始：
＃应该开始：模块
＃必需-停止：
＃应该停止：
＃默认开始：S
＃默认停止：
＃简短描述：用设备节点填充/ dev。
＃说明：在/ dev上挂载tempfs并启动udevd守护程序。
＃设备节点是由udev定义的。
＃X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
```

```
   开始）
      log_info_msg“使用设备节点填充/ dev ...”
      如果！grep -q'[[：space：]] sysfs'/ proc / mounts; 然后
         log_failure_msg2
         msg =“失败：\ n \ n无法创建“
         msg =“ $ {msg}没有SysFS文件系统的设备\ n \ n”
         msg =“ $ {msg}按Enter键后，此系统“
         msg =“ $ {msg}将被停止并关闭电源。\ n \ n”
         log_info_msg“ $ msg”
         log_info_msg“按Enter继续...”
         wait_for_user
         /etc/rc.d/init.d/停止
      科幻

      ＃启动udev守护程序以持续监视并采取行动，
      ＃事件
      / sbin / udevd-守护程序

      ＃现在遍历/ sys以便“冷插”具有以下功能的设备
      ＃已经被发现
      / sbin / udevadm触发器--action = add --type = subsystems
      / sbin / udevadm触发器--action = add --type = devices
      / sbin / udevadm触发器--action = change --type = devices

      ＃现在等待udevd处理我们触发的uevent
      如果！is_true“ $ OMIT_UDEV_SETTLE”; 然后
         / sbin / udevadm解决
      科幻

      ＃如果系统上有任何基于LVM的分区，请确保它们
      ＃已激活，因此可以使用它们。
      如果[-x / sbin / vgchange]; 然后/ sbin / vgchange -ay> / dev / null; 科幻

      log_success_msg2
      ;;

   *）
      回声“用法$ {0} {开始}”
      1号出口
      ;;
埃萨克

出口0

＃结束udev
```

D.6。/etc/rc.d/init.d/swap

```
#！/ bin / sh
################################################ ######################
#开始交换
#
#说明：交换控制脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
################################################ ######################

###开始初始化信息
#提供：交换
#必填-开始：udev
#应该开始：模块
#必需-停止：localnet
#应该停止：
#默认开始：S
#默认停止：0 6
#简短描述：挂载和卸载交换分区。
#说明：装载和卸载在中定义的交换分区
#/ etc / fstab。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      log_info_msg“正在激活所有交换文件/分区...”
      交换-a
      valuate_retval
      ;;

   停）
      log_info_msg“正在取消所有交换文件/分区...”
      交换-a
      valuate_retval
      ;;
```

```
    重新开始）
        $ {0}止损
        睡觉1
        $ {0}开始
        ;;

    状态）
        log_success_msg“正在检索交换状态。”
        交换子-s
        ;;

    *）
        echo“用法：$ {0} {开始|停止|重新启动|状态}”
        1号出口
        ;;
埃萨克

出口0

#结束交换
```

D.7。/etc/rc.d/init.d/setclock

```
#！/ bin / sh
########################################################################
#开始设置时钟
#
#描述：设置Linux Clock
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
########################################################################

###开始初始化信息
#提供：
#必选-开始：
#应该开始：模块
#必需-停止：
#应该停止：$ syslog
#默认开始：S
```

```
＃默认停止：
＃简短描述：存储和恢复硬件时钟中的时间
＃说明：引导时，系统时间是从hwclock获得的。的
＃硬件时钟也可以在关机时设置。
＃X-LFS提供-通过：LFS BLFS
###结束初始化信息

。/ lib / lsb / init-functions

[-r / etc / sysconfig / clock] &&。/ etc / sysconfig /时钟


大小写“ $ {UTC}”
   是|是| 1）
      CLOCKPARAMS =“ $ {CLOCKPARAMS} --utc”
      ;;

   否|假| 0）
      CLOCKPARAMS =“ $ {CLOCKPARAMS} --localtime”
      ;;

埃萨克

案例$ {1}在
   开始）
      hwclock --hctosys $ {CLOCKPARAMS}> / dev / null
      ;;

   停）
      log_info_msg“正在设置硬件时钟...”
      hwclock --systohc $ {CLOCKPARAMS}> / dev / null
      valuate_retval
      ;;

   *）
      回声“用法：$ {0} {开始|停止}”
      1号出口
      ;;

埃萨克

出口0
```

D.8。/etc/rc.d/init.d/checkfs

```
#! / bin / sh
################################################## ######################
#开始checkfs
#
#说明：文件系统检查
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#A.吕贝克-luebke@users.sourceforge.net
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
#基于LFS-3.1和更早版本的checkfs脚本。
#
#来自man fsck
#0-没有错误
#1-文件系统错误已更正
#2-系统应重新启动
#4-文件系统错误未得到纠正
#8-操作错误
#16-使用或语法错误
#32-通过用户请求取消Fsck
#128-共享库错误
#
################################################## ######################

###开始初始化信息
#提供：checkfs
#必选-开始：udev swap $ time
#应该开始：
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：
#简短描述：挂载之前检查本地文件系统。
#说明：挂载前检查本地文件系统。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      如果[-f / fastboot]; 然后
```

```
   msg =“ / fastboot找到，将省略”
   msg =“ $ {msg}文件系统根据要求进行检查。\ n”
   log_info_msg“ $ {msg}”
   出口0
科幻

log_info_msg“正在以只读模式安装根文件系统...”
挂载-n -o重新挂载，ro /> / dev / null

如果[$ {? }！= 0]; 然后
   log_failure_msg2
   msg =“ \ n \ n无法检查根目录”
   msg =“ $ {msg}文件系统，因为无法挂载”
   msg =“ $ {msg}处于只读模式。\ n \ n”
   msg =“ $ {msg}按Enter键后，该系统将为“
   msg =“ $ {msg}已停止并关闭电源。\ n \ n”
   log_failure_msg“ $ {msg}”

   log_info_msg“按Enter继续...”
   wait_for_user
   /etc/rc.d/init.d/停止
其他
   log_success_msg2
科幻

如果[-f / forcefsck]; 然后
   msg =“ / forcefsck找到，强制文件”
   msg =“ $ {msg}系统根据要求进行检查。”
   log_success_msg“ $ msg”
   options =“-f”
其他
   options =“”
科幻

log_info_msg“正在检查文件系统...”
＃注意：-a选项曾经是-p; 但这失败，例如在fsck.minix上
如果is_true“ $ VERBOSE_FSCK”; 然后
  fsck $ {选项} -a -A -C -T
其他
  fsck $ {选项} -a -A -C -T> / dev / null
科幻

error_value = $ {? }

如果[“ $ {error_value}” = 0]; 然后
   log_success_msg2
```

```
科幻

如果[“ $ {error_value}” = 1]; 然后
   msg =“ \ n警告：\ n \ n文件系统错误”
   msg =“ $ {msg}已找到并已更正。\ n”
   msg =“ $ {msg}您可能需要仔细检查“
   msg =“ $ {msg}所有内容均已正确修复。”
   log_warning_msg“ $ msg”
科幻

如果[“ $ {error_value}” = 2 -o“ $ {error_value}” = 3]; 然后
   msg =“ \ n警告：\ n \ n文件系统错误”
   已找到msg =“ $ {msg}，并且已被“
   msg =“ $ {msg}已更正，但“
   msg =“ $ {msg}错误必须重新启动此系统。\ n \ n”
   msg =“ $ {msg}按Enter键后，
   msg =“ $ {msg}此系统将重新启动\ n \ n”
   log_failure_msg“ $ msg”

   log_info_msg“按Enter继续...”
   wait_for_user
   重启-f
科幻

if [“ $ {error_value}” -gt 3 -a“ $ {error_value}” -lt 16]; 然后
   msg =“ \ nFAILURE：\ n \ n文件系统错误”
   遇到msg =“ $ {msg}，不能为“
   msg =“ $ {msg}已自动修复。\ n此系统“
   msg =“ $ {msg}无法继续启动，将“
   因此，msg =“ $ {msg}将被暂停，直到那些“
   msg =“ $ {msg}错误由“
   msg =“ $ {msg}系统管理员。\ n \ n”
   msg =“ $ {msg}按Enter键后，该系统将为“
   msg =“ $ {msg}已停止并关闭电源。\ n \ n”
   log_failure_msg“ $ msg”

   log_info_msg“按Enter继续...”
   wait_for_user
   /etc/rc.d/init.d/停止
科幻

如果[“ $ {error_value}” -ge 16]; 然后
   msg =“ FAILURE：\ n \ n意外失败”
   msg =“ $ {msg}正在运行fsck。退出并出现错误“
   msg =“ $ {msg}代码：$ {error_value}。\ n”
   log_info_msg $ msg
```

```
        退出$ {error_value}
    科幻

    出口0
    ;;
  *)
    回声“用法：$ {0} {开始}”
    1号出口
    ;;
埃萨克

#结束checkfs
```

D.9。/etc/rc.d/init.d/mountfs

```
#！/ bin / sh
########################################################################
#开始mountfs
#
#说明：文件系统挂载脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
########################################################################

###开始初始化信息
#提供：$ local_fs
#必选-开始：udev checkfs
#应该开始：
#必需-停止：交换
#应该停止：
#默认开始：S
#默认停止：0 6
#简短描述：挂载/卸载/ etc / fstab中定义的本地文件系统。
#说明：重新装载对根文件系统的读/写并装载所有
#在/ etc / fstab中定义的其余本地文件系统
#开始。以只读方式重新安装根文件系统并进行卸载
#停止中剩余的文件系统。
#X-LFS提供-通过：LFS
###结束初始化信息
```

```
。/ lib / lsb / init-functions

情况为“ $ {1}”
  开始）
    log_info_msg“以读写模式重新安装根文件系统...”
    mount --options remount，rw /> / dev / null
    valuate_retval

    ＃删除与fsck相关的文件系统水印。
    rm -f / fastboot / forcefsck

    ＃确保/ dev / pts存在
    mkdir -p / dev / pts

    ＃这将挂载所有不包含_netdev的文件系统
    ＃他们的选项列表。_netdev表示网络文件系统。

    log_info_msg“正在安装其余文件系统...”
    挂载--all --test-opts no_netdev> / dev / null
    valuate_retval
    退出$ failed
    ;;

  停）
    ＃不要卸载虚拟文件系统，如/ run
    log_info_msg“正在卸载所有其他当前已安装的文件系统...”
    ＃确保删除所有循环恶魔
    失物招领
    umount --all --detach-loop-只读\
           --types notmpfs，nosysfs，nodevtmpfs，noproc，nodevpts> / dev / null
    valuate_retval

    ＃确保/挂载为只读（umount bug）
    mount --options重新安装，ro /

    ＃使所有LVM卷组不可用（如果适用）
    ＃如果swap或/在LVM分区上，则此操作失败
    #if [-x / sbin / vgchange]; 然后/ sbin / vgchange -an> / dev / null; 科幻
    ;;

  *）
    回声“用法：$ {0} {开始|停止}”
    1号出口
    ;;
埃萨克
```

```
#结束mountfs
```

D.10。/etc/rc.d/init.d/udev_retry

```
#！/ bin / sh
########################################################## ######################
#开始udev_retry
#
#说明：Udev冷插脚本（重试）
#
#作者：亚历山大•E•帕特拉科夫
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#布赖恩•卡兹班-
#
#版本：LFS 7.0
#
########################################################## ######################

###开始初始化信息
#提供：udev_retry
#必填-开始：udev
#应该开始：$ local_fs
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：
#简短描述：重播失败的事件，并创建其他设备。
#说明：重放由于以下原因而跳过的所有失败的事件
#缓慢的硬件初始化，并创建所需的硬件
#个设备节点
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      log_info_msg“重试失败的事件，如果有的话...”

      #从udev-186开始，--run选项不再有效
      #rundir = $（/ sbin / udevadm info --run）
      rundir = /运行/ udev
```

```
      #从Debian：“复制在/挂载之前生成的规则
      #读写”：

      用于$ {rundir} / tmp-rules中的文件-*；做
         dest = $ {file ## * tmp-rules--}
         [“ $ dest” ='*'] &&中断
         cat $ file >> /etc/udev/rules.d/$dest
         rm -f $ file
      做完了

      #重新触发可能失败的事件，
      #希望他们现在能成功
      / bin / sed -e's /#.*$//'/ etc / sysconfig / udev_retry | / bin / grep -v'^ $'| \
      边读行；做
         用于$ line中的子系统；做
            / sbin / udevadm触发器--subsystem-match = $ subsystem --action = add
         做完了
      做完了

      #现在等待udevd处理我们触发的uevent
      如果！is_true“ $ OMIT_UDEV_RETRY_SETTLE”；然后
         / sbin / udevadm解决
      科幻

      valuate_retval
      ;;

   *)
      回声“用法$ {0} {开始}”
      1号出口
      ;;
埃萨克

出口0

#结束udev_retry
```

D.11。/etc/rc.d/init.d/cleanfs

```
#！/ bin / sh
########################################################################
#开始清理
#
#说明：清理文件系统
```

```
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
################################################## ######################

###开始初始化信息
#提供：cleanfs
#必填项-$ local_fs
#应该开始：
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：
#简短描述：在引导过程的早期清理临时目录。
#说明：清理临时目录/ var / run，/ var / lock和
#（可选）/ tmp。cleanfs还会创建/ var / run / utmp
#和/ etc / sysconfig / createfiles中定义的任何文件。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

#在启动时创建文件/目录的功能。
create_files（）
{
   #输入文件描述符9并输出到stdin（重定向）
   exec 9>＆0 </ etc / sysconfig / createfiles

   而读取名称类型为perm usr grp dtype maj min junk
   做
      #忽略注释和空白行。
      大小写为“ $ {name}”
         “” | \#*）继续;;
      埃萨克

      #忽略现有文件。
      如果[！-e“ $ {name}”]; 然后
         #根据类型创建内容。
         大小写“ $ {type}”
            目录）
               mkdir“ $ {name}”
               ;;
```

```
            文件）
               ：>“ $ {name}”
               ;;
            开发）
               大小写“ $ {dtype}”
                  字符）
                     mknod“ $ {name}” c $ {maj} $ {min}
                     ;;
                  块）
                     mknod“ $ {name}” b $ {maj} $ {min}
                     ;;
                  管）
                     mknod“ $ {name}” p
                     ;;
                  *）
                     log_warning_msg“ \ n未知的设备类型：$ {dtype}”
                     ;;
               埃萨克
               ;;
            *）
               log_warning_msg“ \ n未知类型：$ {type}”
               继续
               ;;
         埃萨克

         #也设置权限。
         chown $ {usr}：$ {grp}“ $ {name}”
         chmod $ {perm}“ $ {name}”
      科幻
   做完了

   #关闭文件描述符9（结束重定向）
   exec 0>&9 9>&-
   返回0
}

情况为“ $ {1}”
   开始）
      log_info_msg“正在清理文件系统：”

      如果[“ $ {SKIPTMPCLEAN}” =“”]；然后
         log_info_msg2“ / tmp”
         cd / tmp &&
         找 。-xdev -mindepth 1！名称丢失+找到-删除|| 失败= 1
      科幻
```

```
      > / var / run / utmp

      如果grep -q'^ utmp：'/ etc / group; 然后
         chmod 664 / var / run / utmp
         chgrp utmp / var / run / utmp
      科幻

      （退出$ {failed}）
      valuate_retval

      如果egrep -qv'^（#| $）'/ etc / sysconfig / createfiles 2> / dev / null; 然后
         log_info_msg“正在创建文件和目录...”
         create_files＃总是返回0
         valuate_retval
      科幻

      退出$ failed
      ;;
   *）
      回声“用法：$ {0} {开始}”
      1号出口
      ;;
埃萨克

＃结束清洁
```

D.12。/etc/rc.d/init.d/console

```
＃！/ bin / sh
################################################## ######################
＃开始控制台
＃
＃描述：设置键盘映射和屏幕字体
＃
＃作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
＃亚历山大•E•帕特拉科夫
＃DJ Lucas-dj AT linuxfromscratch DOT org
＃更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
＃
＃版本：LFS 7.0
＃
################################################## ######################

###开始初始化信息
```

```
＃提供：控制台
＃必选-开始：
＃应该开始：$ local_fs
＃必需-停止：
＃应该停止：
＃默认开始：S
＃默认停止：
＃简短说明：设置本地化的控制台。
＃说明：为用户的字体设置字体和语言设置
＃local由/ etc / sysconfig / console定义。
＃X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

＃以英语为母语的人可能根本没有/ etc / sysconfig / console
[-r / etc / sysconfig / console] &&。/ etc / sysconfig /控制台

是真的（）
{
   [“ $ 1” =“ 1”] || [“ $ 1” =“是”] || [“ $ 1” =“ true”]
}

失败= 0

情况为“ $ {1}”
   开始）
      ＃看看我们是否需要做任何事情
      如果[-z“ $ {KEYMAP}”] && [-z“ $ {KEYMAP_CORRECTIONS}”]] &&
         [-z“ $ {FONT}”] && [-z“ $ {LEGACY_CHARSET}”] &&
         ！is_true“ $ {UNICODE}”; 然后
         出口0
      科幻

      ＃在这条线以下应该没有假故障！
      log_info_msg“正在设置Linux控制台...”

      ＃找出是否使用了帧缓冲控制台
      [-d / sys / class / graphics / fb0] && use_fb = 1 || use_fb = 0

      ＃找出将控制台设置为
      ＃所需模式
      is_true“ $ {UNICODE}” &&
         MODE_COMMAND =“ echo -en’\ 033％G’&& kbd_mode -u” ||
         MODE_COMMAND =“ echo -en’\ 033％@ \ 033（K’&& kbd_mode -a”
```

```
    #在framebuffer控制台上，必须为每个vt设置字体
    #UTF-8模式。在非UTF-8模式下也不会受到伤害。

    ! is_true“ $ {use_fb}” || [-z“ $ {FONT}”] ||
       MODE_COMMAND =“ $ {MODE_COMMAND} && setfont $ {FONT}”

    #将该命令应用于提到的所有控制台
    #/ etc / inittab。重要提示：在UTF-8模式下，
    #发生在setfont之前，否则将发生内核错误
    #显示，字体的unicode映射将不会
    # 用过的。

    用于`grep'^ [^#]。中的TTY。* respawn：/ sbin / agetty'/ etc / inittab |
       grep -o'\ btty [[：digit：]] * \ b'`
    做
       openvt -f -w -c $ {TTY#tty}-\
          / bin / sh -c“ $ {MODE_COMMAND}” || 失败= 1
    做完了

    #设置字体（如果上面尚未设置）和键盘映射
    [“ $ {use_fb}” ==“ 1”] || [-z“ $ {FONT}”] || setfont $ FONT || 失败= 1

    [-z“ $ {KEYMAP}”] ||
       loadkeys $ {KEYMAP}> / dev / null 2>&1 ||
       失败= 1

    [-z“ $ {KEYMAP_CORRECTIONS}”] ||
       loadkeys $ {KEYMAP_CORRECTIONS}> / dev / null 2>&1 ||
       失败= 1

    #将键盘映射从$ LEGACY_CHARSET转换为UTF-8
    [-z“ $ LEGACY_CHARSET”] ||
       dumpkeys -c“ $ LEGACY_CHARSET” | loadkeys -u> / dev / null 2>&1 ||
       失败= 1

    #如果以上任何命令失败，则位于
    #top将$ failed设置为1
    （退出$ failed）
    valuate_retval

    退出$ failed
    ;;

*）
    回声“用法：$ {0} {开始}”
    1号出口
```

```
      ;;
埃萨克

#终端控制台
```

D.13。/etc/rc.d/init.d/localnet

```
#！/ bin / sh
############################################### ######################
#开始localnet
#
#说明：环回设备
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
############################################### ######################

###开始初始化信息
#提供：localnet
#必填项-$ local_fs
#应该开始：
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：0 6
#简短描述：启动本地网络。
#说明：设置计算机的主机名并启动
#回送接口。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions
[-r / etc / sysconfig / network] &&。/ etc / sysconfig / network
[-r / etc / hostname] && HOSTNAME =`cat / etc / hostname`

情况为“ $ {1}”
   开始）
      log_info_msg“打开回送接口...”
      ip addr添加127.0.0.1/8标签lo dev lo
      IP链接设置
```

```
      valuate_retval

      log_info_msg“将主机名设置为$ {HOSTNAME} ...”
      主机名$ {HOSTNAME}
      valuate_retval
      ;;

   停）
      log_info_msg“关闭回送接口...”
      IP链接设置为低
      valuate_retval
      ;;

   重新开始）
      $ {0}止损
      睡觉1
      $ {0}开始
      ;;

   状态）
      回声“主机名是：$（主机名）”
      ip链接显示lo
      ;;

   *）
      echo“用法：$ {0} {开始|停止|重新启动|状态}”
      1号出口
      ;;
埃萨克

出口0

#结束本地网
```

D.14。/etc/rc.d/init.d/sysctl

```
#！/ bin / sh
################################################## ######################
#开始sysctl
#
#说明：文件使用/etc/sysctl.conf设置内核运行时
#个参数
#
#作者：内森·库尔森（Nathan AT linuxfromscratch DOT org）
```

```
#Matthew Burgress（马修AT linuxfromscratch DOT组织）
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
################################################## ######################

###开始初始化信息
#提供：sysctl
#必需-开始：mountvirtfs
#应该开始：
#必需-停止：
#应该停止：
#默认开始：S
#默认停止：
#简短描述：对proc文件系统进行更改
#说明：根据proc文件系统中的定义进行更改
#/etc/sysctl.conf。参见“ man sysctl（8）”。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      如果[-f“ /etc/sysctl.conf”]; 然后
         log_info_msg“正在设置内核运行时参数...”
         sysctl -q -p
         valuate_retval
      科幻
      ;;

   状态）
      sysctl -a
      ;;

   *）
      回声“用法：$ {0} {开始|状态}”
      1号出口
      ;;
埃萨克

出口0

#结束sysctl
```

D.15。/etc/rc.d/init.d/sysklogd

```
#！/ bin / sh
########################################### ######################
#开始sysklogd
#
#说明：Sysklogd加载程序
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
########################################### ######################

###开始初始化信息
#提供：$ syslog
#必填-开始：localnet
#应该开始：
#必需-停止：$ local_fs sendsignals
#应该停止：
#默认开始：3 4 5
#默认停止：0 1 2 6
#简短描述：启动内核和系统日志守护程序。
#说明：启动内核和系统日志守护程序。
#/ etc / fstab。
#X-LFS提供-通过：LFS
###结束初始化信息

#注意：由于可能，sysklogd未在运行级别2中启动
#远程日志记录配置

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      log_info_msg“正在启动系统日志守护程序...”
      parms = $ {SYSKLOGD_PARMS -’- m 0’}
      start_daemon / sbin / syslogd $ parms
      valuate_retval

      log_info_msg“正在启动内核日志守护程序...”
```

```
      start_daemon / sbin / klogd
      valuate_retval
      ;;

   停）
      log_info_msg“正在停止内核日志守护程序...”
      killproc / sbin / klogd
      valuate_retval

      log_info_msg“正在停止系统日志守护程序...”
      killproc / sbin / syslogd
      valuate_retval
      ;;

   重新加载）
      log_info_msg“正在重新加载系统日志守护程序配置文件...”
      pid =`pidofproc syslogd`
      kill -HUP“ $ {pid}”
      valuate_retval
      ;;

   重新开始）
      $ {0}止损
      睡觉1
      $ {0}开始
      ;;

   状态）
      statusproc / sbin / syslogd
      statusproc博客
      ;;

   *）
      回声“用法：$ {0} {开始|停止|重新加载|重新启动|状态}”
      1号出口
      ;;
埃萨克

出口0

＃结束sysklogd
```

D.16。/etc/rc.d/init.d/network

```
#! / bin / sh
############################################### ######################
#开始网络
#
#说明：网络控制脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#Nathan Coulson-Nathan AT linuxfromscratch DOT org
#Kevin P.Fleming-kpfleming@linuxfromscratch.org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
############################################### ######################

###开始初始化信息
#提供：$ network
#必选-开始：$ local_fs swap localnet
#应该开始：$ syslog
#必需-停止：$ local_fs交换localnet
#应该停止：$ syslog
#默认开始：3 4 5
#默认停止：0 1 2 6
#简短描述：启动和配置网络接口。
#说明：启动和配置网络接口。
#X-LFS提供-通过：LFS
###结束初始化信息

情况为“ $ {1}”
   开始）
      #启动所有网络接口
      用于/etc/sysconfig/ifconfig.*中的文件
      做
         interface = $ {file ## * / ifconfig。}

         #如果$ file为*，则跳过（因为未找到任何内容）
         如果[“ $ {interface}” =“ *”]
         然后
            继续
         科幻

         / sbin / ifup $ {interface}
      做完了
      ;;
```

```
    停）
        #卸载任何网络安装的文件系统
         umount --all --force --types nfs，cifs，nfs4

        #反向清单
        net_files =“”
        用于/etc/sysconfig/ifconfig.*中的文件
        做
            net_files =“ $ {file} $ {net_files}”
        做完了

        #停止所有网络接口
        用于$ {net_files}中的文件
        做
            interface = $ {file ## * / ifconfig。}

            #如果$ file为*，则跳过（因为未找到任何内容）
            如果[“ $ {interface}” =“ *”]
            然后
                继续
            科幻

            / sbin / ifdown $ {interface}
        做完了
        ;;

    重新开始）
        $ {0}止损
        睡觉1
        $ {0}开始
        ;;

    *）
        回声“用法：$ {0} {开始|停止|重新启动}”
        1号出口
        ;;
埃萨克

出口0

#终端网络
```

D.17。/etc/rc.d/init.d/sendsignals

```
#！/ bin / sh
############################################### ######################
#开始发送信号
#
#说明：Sendsignals脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
############################################### ######################

###开始初始化信息
#提供：sendsignals
#必选-开始：
#应该开始：
#必需-停止：$ local_fs交换localnet
#应该停止：
#默认开始：
#默认停止：0 6
#简短描述：尝试杀死其余进程。
#说明：尝试杀死其余进程。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   停）
      log_info_msg“正在发送所有进程TERM信号...”
      killall5 -15
      error_value = $ {? }

      睡眠$ {KILLDELAY}

      如果[“ $ {error_value}” = 0 -o“ $ {error_value}” = 2]; 然后
         log_success_msg
      其他
         log_failure_msg
      科幻

      log_info_msg“正在发送所有进程KILL信号...”
      killall5 -9
```

```
      error_value = $ {? }

      睡眠$ {KILLDELAY}

      如果[“ $ {error_value}” = 0 -o“ $ {error_value}” = 2]; 然后
         log_success_msg
      其他
         log_failure_msg
      科幻
      ;;

   *)
      回声“用法：$ {0} {停止}”
      1号出口
      ;;

埃萨克

出口0

#结束发送信号
```

D.18。/etc/rc.d/init.d/reboot

```
#！/ bin / sh
########################################################################
#开始重启
#
#说明：重新启动脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
########################################################################

###开始初始化信息
#提供：重新启动
#必选-开始：
#应该开始：
#必需-停止：
#应该停止：
```

```
#默认开始：6
#默认停止：
#简短描述：重新引导系统。
#说明：重新引导系统。
#X-LFS提供-通过：LFS
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   停）
      log_info_msg“正在重启系统...”
      重新启动-d -f -i
      ;;

   *）
      回声“用法：$ {0} {停止}”
      1号出口
      ;;

埃萨克

#重启结束
```

D.19。/etc/rc.d/init.d/halt

```
#！/ bin / sh
########################################################################
#开始停止
#
#说明：暂停脚本
#
#作者：Gerard Beekmans-Gerard AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
########################################################################

###开始初始化信息
#提供：停止
#必选-开始：
#应该开始：
```

```
#必需-停止：
#应该停止：
#默认开始：0
#默认停止：
#简短描述：停止系统。
#说明：停止系统。
#X-LFS提供-通过：LFS
###结束初始化信息

情况为“ $ {1}”
   停）
      停止-d -f -i -p
      ;;

   *）
      回声“用法：{停止}”
      1号出口
      ;;
埃萨克

#停止
```

D.20。/etc/rc.d/init.d/template

```
#！/ bin / sh
############################################### ######################
#开始脚本名
#
#说明：
#
#作者：
#
#版本：LFS xx
#
#注意：
#
############################################### ######################

###开始初始化信息
#提供：模板
#必选-开始：
#应该开始：
#必需-停止：
#应该停止：
```

```
#默认开始：
#默认停止：
#  简短的介绍：
#说明：
#X-LFS提供：
###结束初始化信息

。/ lib / lsb / init-functions

情况为“ $ {1}”
   开始）
      log_info_msg“正在启动...”
      start_daemon fully_qualified_path
      ;;

   停）
      log_info_msg“正在停止...”
      killproc fully_qualified_path
      ;;

   重新开始）
      $ {0}止损
      睡觉1
      $ {0}开始
      ;;

   *）
      回声“用法：$ {0} {开始|停止|重新启动}”
      1号出口
      ;;
埃萨克

出口0

#结束脚本名
```

D.21。/ etc / sysconfig /模块

```
################################################ ######################
#开始/ etc / sysconfig / modules
#
#说明：模块自动加载配置
#
#作者：
```

```
#
#版本：00.00
#
#注意：此文件的语法如下：
#<模块> [<arg1> <arg2> ...]
#
#每个模块应该在自己的行上，并且需要任何选项
传递给模块的#应该跟随它。行分隔符为
#一个空格或一个制表符。
################################################ ######################

#结束/ etc / sysconfig / modules
```

D.22。/ etc / sysconfig / createfiles

```
################################################ ######################
#开始/ etc / sysconfig / createfiles
#
#说明：Createfiles脚本配置文件
#
#作者：
#
#版本：00.00
#
#注意：此文件的语法如下：
#如果类型等于“文件”或“目录”
#<文件名> <类型> <权限> <用户> <组>
#如果类型等于“ dev”
#<文件名> <类型> <权限> <用户> <组> <设备类型>
#<主要> <次要>
#
#<filename>是要创建的文件的名称
#<type>是file，dir或dev。
#file创建一个新文件
#dir创建一个新目录
#dev创建一个新设备
#<devtype>是块，字符或管道
#block创建一个块设备
#char创建一个字符设备
#pipe创建一个管道，这将忽略<major>和
#<minor>字段
#<major>和<minor>是用于
# 装置。
################################################ ######################
```

```
#结束/ etc / sysconfig / createfiles
```

D.23。/ etc / sysconfig / udev-retry

```
################################################ ######################
#开始/ etc / sysconfig / udev_retry
#
#说明：udev_retry脚本配置
#
#作者：
#
#版本：00.00
#
#注意：mountfs之后可能需要重新触发的每个子系统
#运行次数应在此文件中列出。可能的子系统是
#此处列出的是rtc（由于/ var / lib / hwclock / adjtime）和声音
#（由于/var/lib/alsa/asound.state和/ usr / sbin / alsactl）。
#条目之间用空格分隔。
################################################ ######################

rtc

#结束/ etc / sysconfig / udev_retry
```

D.24。/ sbin / ifup

```
#！/ bin / sh
################################################ ######################
#开始/ sbin / ifup
#
#说明：界面向上
#
#作者：Nathan Coulson-Nathan AT linuxfromscratch DOT org
#Kevin P.Fleming-kpfleming@linuxfromscratch.org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#DJ Lucas-dj AT linuxfromscratch DOT org
#
#版本：LFS 7.7
#
#注意：IFCONFIG变量传递到SERVICE脚本
/ lib / services目录中的#，以指示
#service应该作为获取接口规范的来源。
```

```
#
############################################### ####################

up ()
{
  log_info_msg“启动$ {1}接口...”

  如果ip链接显示$ 1> / dev / null 2>&1; 然后
     link_status =`ip链接显示$ 1`

     如果[-n“ $ {link_status}”]; 然后
        如果！回声“ $ {link_status}” | grep -q UP; 然后
           ip链接设置$ 1
        科幻
     科幻

  其他
     log_failure_msg“接口$ {IFACE}不存在。”
     1号出口
  科幻

  valuate_retval
}

RELEASE =“ 7.7”

USAGE =“用法：$ 0 [-hV] [--help] [--version]接口”
VERSTR =“ LFS ifup，版本$ {RELEASE}”

而[$#-gt 0]; 做
   案例“ $ 1”在
      -帮助| -h）help =“ y”; 打破

      --version | -V）回显“ $ {VERSTR}”；退出0 ;;

      -*）回显“ ifup：$ {1}：无效选项”>&2
                      回声“ $ {USAGE}”>&2
                      出口2 ;;

      *）打破;;
   埃萨克
做完了

如果[-n“ $ help”]; 然后
   回显“ $ {VERSTR}”
   回显“ $ {USAGE}”
```

```
   回声
   猫<< HERE_EOF
ifup用于启动网络接口。介面
参数，例如eth0或eth0：2，必须与
接口规范文件，例如/etc/sysconfig/ifconfig.eth0:2。

HERE_EOF
   出口0
科幻

file = / etc / sysconfig / ifconfig。$ {1}

＃跳过备份文件
[| $ {file}“ =” $ {file%“”〜“”}“] || 出口0

。/ lib / lsb / init-functions

如果[！-r“ $ {文件}”]；然后
   “ log_failure_msg”无法启动$ {1}界面！$ {file}丢失或无法访问。
   1号出口
科幻

。$文件

如果[“ $ IFACE” =“”]；然后
   “ log_failure_msg”无法启动$ {1}接口！$ {file}没有定义接口[IFACE]。
   1号出口
科幻

＃如果通过引导和ONBOOT启动，请勿处理此服务
＃未设置为是
如果[“ $ {IN_BOOT}” =“ 1” -a“ $ {ONBOOT}”！=“”是“]；然后
   出口0
科幻

＃调出界面
如果[“ $ VIRTINT”！=“是”]；然后
   涨$ {IFACE}
科幻

$ {SERVICE}中的S；做
  如果[！-x“ / lib / services / $ {S}”]]；然后
    MSG =“ \ n无法处理$ {file}。要么是“
    MSG =“ $ {MSG} SERVICE'$ {S}不存在“
    MSG =“ $ {MSG} or或无法执行。”
    log_failure_msg“ $ MSG”
```

```
    1号出口
  科幻
做完了

如果[“ $ {SERVICE}” =“ wpa”]; 然后log_success_msg; 科幻

＃创建/配置接口
$ {SERVICE}中的S；做
  IFCONFIG = $ {file} / lib / services / $ {S} $ {IFACE}起
做完了

＃设置链接虚拟接口
如果[“ $ {VIRTINT}” ==“是”]; 然后
   涨$ {IFACE}
科幻

＃调出任何其他接口组件
为我在$ INTERFACE_COMPONENTS中；加$ I；做完了

＃根据需要设置MTU。检查MTU是否具有“良好”值。
如果测试-n“ $ {MTU}”; 然后
   如果[[$ {MTU} =〜^ [0-9] + $]] && [[$ MTU -ge 68]]; 然后
      为我在$ IFACE $ INTERFACE_COMPONENTS中；做
         ip link set dev $ I mtu $ MTU;
      做完了
   其他
      log_info_msg2“无效的MTU $ MTU”
   科幻
科幻

＃根据需要设置路由默认网关
如果[-n“ $ {GATEWAY}”]; 然后
   如果IP路由| grep -q默认; 然后
      “ log_warning_msg”网关已设置；正在跳过。
   其他
      log_info_msg“正在将默认网关$ {GATEWAY}添加到$ {IFACE}接口...”
      ip route通过$ {GATEWAY} dev $ {IFACE}添加默认值
      valuate_retval
   科幻
科幻

＃结束/ sbin / ifup
```

D.25。/ sbin / ifdown

```
＃！/ bin / bash
################################################ ######################
＃开始/ sbin / ifdown
＃
＃描述：接口关闭
＃
＃作者：Nathan Coulson-Nathan AT linuxfromscratch DOT org
＃Kevin P.Fleming-kpfleming@linuxfromscratch.org
＃更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
＃
＃版本：LFS 7.0
＃
＃注意：IFCONFIG变量传递给找到的脚本
/ lib / services目录中的＃，以指示
＃service应该作为获取接口规范的来源。
＃
################################################ ######################

RELEASE =“ 7.0”

USAGE =“用法：$ 0 [-hV] [--help] [--version]接口”
VERSTR =“ LFS ifdown，版本$ {RELEASE}”

而[$＃-gt 0]; 做
   案例“ $ 1”在
      -帮助| -h）help =“ y”; 打破

      --version | -V）回显“ $ {VERSTR}”；退出0 ;;

      -*）回显“ ifup：$ {1}：无效选项”>＆2
                      回声“ $ {USAGE}”>＆2
                      出口2 ;;

      *）打破;;
   埃萨克
做完了

如果[-n“ $ help”]; 然后
   回显“ $ {VERSTR}”
   回显“ $ {USAGE}”
   回声
   猫<< HERE_EOF
ifdown用于关闭网络接口。介面
参数，例如eth0或eth0：2，必须与
接口规范文件，例如/etc/sysconfig/ifconfig.eth0:2。
```

```
HERE_EOF
   出口0
科幻

file = / etc / sysconfig / ifconfig。$ {1}

＃跳过备份文件
[| $ {file}“ =” $ {file%“”∽“”}“] || 出口0

。/ lib / lsb / init-functions

如果[! -r“ $ {文件}”]; 然后
   log_warning_msg“ $ {文件}丢失或无法访问。”
   1号出口
科幻

。$ {file}

如果[“ $ IFACE” =“”]; 然后
    “ log_failure_msg” $ {文件}没有定义接口[IFACE]。
   1号出口
科幻

＃我们只需要首先服务即可关闭该界面
S =`echo $ {SERVICE} | 切-f1 -d“”`

如果ip链接显示$ {IFACE}> / dev / null 2>＆1; 然后
   如果[-n“ $ {S}” -a -x“ / lib / services / $ {S}”]; 然后
     IFCONFIG = $ {文件} / lib / services / $ {S} $ {IFACE}下
   其他
     MSG =“无法处理$ {file}。要么是”
     MSG =“ $ {MSG} SERVICE变量未设置”
     MSG =“ $ {MSG}或指定的服务无法执行。”
     log_failure_msg“ $ MSG”
     1号出口
  科幻
其他
   log_warning_msg“接口$ {1}不存在。”
科幻

＃如果设备中还有其他接口，请保留接口状态
link_status =`ip链接显示$ {IFACE} 2> / dev / null`

如果[-n“ $ {link_status}”]; 然后
   如果[“ $（echo” $ {link_status}“ | grep UP）”！=“”]; 然后
```

```
      如果[“ $（ip addr show $ {IFACE} | grep'inet'）” ==“”]; 然后
         log_info_msg“关闭$ {IFACE}接口...”
         ip链接设置为$ {IFACE}
         valuate_retval
      科幻
   科幻
科幻

#结束/ sbin / ifdown
```

D.26。/ lib / services / ipv4-static

```
#！/ bin / sh
################################################ ######################
#开始/ lib / services / ipv4-static
#
#说明：IPV4静态启动脚本
#
#作者：Nathan Coulson-Nathan AT linuxfromscratch DOT org
#Kevin P.Fleming-kpfleming@linuxfromscratch.org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
################################################ ######################

。/ lib / lsb / init-functions
。$ {IFCONFIG}

如果[-z“ $ {IP}”]; 然后
   “ log_failure_msg” \ n $$ {IFCONFIG}中缺少IP变量，无法继续。
   1号出口
科幻

如果[-z“ $ {PREFIX}” -a -z“ $ {PEER}”]; 然后
   “ log_warning_msg” \ nPREFIX变量从$ {IFCONFIG}中丢失，假设为24。
   前缀= 24
   args =“ $ {args} $ {IP} / $ {PREFIX}”

elif [-n“ $ {PREFIX}” -a -n“ $ {PEER}”]; 然后
   “ log_failure_msg”在$ {IFCONFIG}中指定的\ nPREFIX和PEER无法继续。
   1号出口

elif [-n“ $ {PREFIX}”]; 然后
```

```
   args =“ $ {args} $ {IP} / $ {PREFIX}”

elif [-n“ $ {PEER}”]; 然后
   args =“ $ {args} $ {IP}对等$ {PEER}”
科幻

如果[-n“ $ {LABEL}”]; 然后
   args =“ $ {args}标签$ {LABEL}”
科幻

如果[-n“ $ {BROADCAST}”]; 然后
   args =“ $ {args} broadcast $ {BROADCAST}”
科幻

大小写“ $ {2}”
   上）
      如果[“ $（ip addr show $ {1} 2> / dev / null | grep $ {IP} /）” =“”]; 然后
         log_info_msg“正在将IPv4地址$ {IP}添加到$ {1}接口...”
         ip addr add $ {args} dev $ {1}
         valuate_retval
      其他
          “ log_warning_msg”无法将IPv4地址$ {IP}添加到$ {1}。已经存在。
      科幻
   ;;

   下）
      如果[“ $（ip addr show $ {1} 2> / dev / null | grep $ {IP} /）”！=“”]; 然后
         log_info_msg“正在从$ {1}接口中删除IPv4地址$ {IP} ...”
         ip addr del $ {args} dev $ {1}
         valuate_retval
      科幻

      如果[-n“ $ {GATEWAY}”]; 然后
         ＃仅在没有剩余的ipv4地址的情况下删除网关
         如果[“ $（ip addr show $ {1} 2> / dev / null | grep'inet'）”！=“”]; 然后
            log_info_msg“正在删除默认网关...”
            ip route del默认
            valuate_retval
         科幻
      科幻
   ;;

   *）
      回声“用法：$ {0} [interface] {up | down}”
      1号出口
   ;;
```

```
埃萨克

#结束/ lib / services / ipv4-static
```

D.27。/ lib / services / ipv4-static-route

```
#！/ bin / sh
################################################ ######################
#开始/ lib / services / ipv4-static-route
#
#说明：IPV4静态路由脚本
#
#作者：Kevin P. Fleming-kpfleming@linuxfromscratch.org
#DJ Lucas-dj AT linuxfromscratch DOT org
#更新：Bruce Dubbs-bdubbs AT linuxfromscratch DOT org
#
#版本：LFS 7.0
#
################################################ ######################

。/ lib / lsb / init-functions
。$ {IFCONFIG}

大小写为“ $ {TYPE}”
    （“” |“网络”）
      need_ip = 1
      need_gateway = 1
   ;;

    （“默认”）
      need_gateway = 1
      args =“ $ {args}默认”
      desc =“默认”
   ;;

    （“主办”）
      need_ip = 1
   ;;

    （“无法访问”）
      need_ip = 1
      args =“ $ {args}无法访问”
      desc =“ unreachable”
   ;;
```

```
    （*）
       “ log_failure_msg” $ {IFCONFIG}中的未知路由类型（$ {TYPE}）无法继续。
      1号出口
    ;;
埃萨克

如果[-n“ $ {GATEWAY}”]; 然后
   MSG =“无法在$ {IFCONFIG}中为静态路由设置GATEWAY变量。\ n”
   log_failure_msg“ $ MSG仅使用STATIC_GATEWAY，无法继续”
   1号出口
科幻

如果[-n“ $ {need_ip}”]; 然后
   如果[-z“ $ {IP}”]; 然后
      “ log_failure_msg” $ {IFCONFIG}中缺少IP变量，无法继续。
      1号出口
   科幻

   如果[-z“ $ {PREFIX}”]; 然后
      “ log_failure_msg” $ {IFCONFIG}中缺少PREFIX变量，无法继续。
      1号出口
   科幻

   args =“ $ {args} $ {IP} / $ {PREFIX}”
   desc =“ $ {desc} $ {IP} / $ {PREFIX}”
科幻

如果[-n“ $ {need_gateway}”]; 然后
   如果[-z“ $ {STATIC_GATEWAY}”]; 然后
      “ log_failure_msg” $ {IFCONFIG}中缺少STATIC_GATEWAY变量，无法继续。
      1号出口
   科幻
   args =“ $ {args}通过$ {STATIC_GATEWAY}”
科幻

如果[-n“ $ {SOURCE}”]; 然后
       args =“ $ {args} src $ {SOURCE}”
科幻

大小写“ $ {2}”
   上）
      log_info_msg“将$$ {desc}’路由添加到$ {1}接口...”
      ip route添加$ {args} dev $ {1}
      valuate_retval
   ;;
```

```
    下）
       log_info_msg“正在从$ {1}接口中删除'$ {desc}'路由...”
       ip route del $ {args} dev $ {1}
       valuate_retval
    ;;

    *）
       回声“用法：$ {0} [interface] {up | down}”
       1号出口
    ;;
埃萨克

＃结束/ lib / services / ipv4-static-route
```

附录E.Udev配置规则

为方便起见，列出了本附录中的规则。通常按照第6.77节“ Eudev-3.2.8”中的说明进行安装。

E.1。55-lfs.rules

```
＃/etc/udev/rules.d/55-lfs.rules：LFS的规则定义。

＃核心内核设备

＃这使/ dev / rtc可用时立即设置系统时钟。
SUBSYSTEM ==“ rtc”，ACTION ==“ add”，MODE =“ 0644”，RUN + =“ / etc / rc.d / init.d / setclock start”
KERNEL ==“ rtc”，ACTION ==“ add”，MODE =“ 0644”，RUN + =“ / etc / rc.d / init.d / setclock start”

＃通讯设备

KERNEL ==“ ippp [0-9] *”，GROUP =“ dialout”
KERNEL ==“ isdn [0-9] *”，GROUP =“ dialout”
KERNEL ==“ isdnctrl [0-9] *”，GROUP =“ dialout”
KERNEL ==“ dcbri [0-9] *”，GROUP =“ dialout”
```

附录F. LFS许可证

本书已根据知识共享署名-非商业性-相同方式共享2.0许可进行了许可。

可以根据MIT许可从书中提取计算机指令。

F.1。知识共享许可

知识共享法律法规

Attribution-NonCommercial-ShareAlike 2.0

重要

CREATIVE COMMONS CORPORATION不是一家法律公司，也不提供法律服务。分发此许可证不会建立律师与客户之间的关系。创意交流会按“原样”提供此信息。创意委员会对所提供的信息不做任何保证，并且对因使用信息而造成的损害不承担任何责任。

执照

该工作（如下所定义）是根据本创造性公告公共许可（“ CCPL”或“许可”）的条款提供的。该作品受版权和/或其他适用法律的保护。除本许可或版权法授权的范围外，禁止对本作品进行任何其他使用。

行使此处提供的任何工作权利，即表示您接受并同意受本许可条款的约束。考虑到您接受这些条款和条件，许可方授予您此处包含的权利。

1. 定义
   a. “集体作品”是指诸如期刊，选集或百科全书之类的作品，其中未经修改的完整形式的作品，连同其本身构成单独和独立作品的许多其他贡献，被组合成一个集体整个。就本许可而言，构成集体作品的作品将不被视为衍生作品（定义如下）。
   b. “衍生作品”是指基于该作品或基于该作品和其他先前存在的作品的作品，例如翻译，音乐编排，戏剧化，虚构，电影版本，录音，艺术品复制，删节，缩编或任何其他形式的作品可以重铸，变换或改编该作品的其他形式，但就本许可而言，构成集体作品的作品将不被视为衍生作品。为避免疑义，
   c. “许可人”是指根据本许可条款提供作品的个人或实体。
   d. “原始作者”是指创建作品的个人或实体。
   e. “作品”是指根据本许可条款提供的版权著作权作品。
   f. “您”是指根据本许可行使权利的个人或实体，该个人或实体先前未违反本许可有关作品的条款，或者尽管先前曾违反但已获得许可方的明确许可以行使本许可下的权利。
   g. “许可元素”是指由许可方选择并在本许可的标题中指示的以下高级许可属性：出处，非商业性，ShareAlike。
2. 合理使用权。本许可证中的任何内容均无意于减少，限制或限制因版权法或其他适用法律根据合理使用，首次出售或对版权所有者专有权的其他限制而产生的任何权利。

3. 许可授予。在遵守本许可的条款和条件的前提下，许可方特此授予您全球性的，免版税的，非排他的，永久性的（在适用版权期限内）许可，以如下所述行使作品中的权利：

   a. 复制作品，将作品合并到一个或多个集体作品中，并复制纳入集体作品中的作品；

   b. 创作和复制衍生作品；

   c. 通过数字音频传输分发作品的副本或录音制品，公开展示，公开表演和公开表演，包括并入集体作品中的作品；

   d. 通过数字音频传输衍生作品分发，复制，录音，公开展示，公开表演和公开表演；

   无论是现在知道还是以后设计，上述权利均可在所有媒体和格式中行使。上述权利包括进行修改以行使其他媒体和格式的权利所必需的权利。特此保留未经许可方明确授予的所有权利，包括但不限于第4（e）和4（f）节中规定的权利。

4. 限制：以上第3节中授予的许可明确受以下限制限制：

   a. 您只能根据本许可的条款分发，公开展示，公开表演或以数字方式公开执行作品，并且您在作品的每个副本或录音中都必须包括本许可的副本或统一资源标识符。分发，公开展示，公开表演或公开进行数字表演。您不得在作品上提供或施加任何条款来更改或限制本许可的条款或接受者行使本协议授予的权利。您可能没有对作品进行再授权。您必须保留所有引用本许可和免责声明的声明。您不得以与本许可协议的条款相抵触的方式，以任何控制作品访问或使用的技术措施，分发，公开展示，公开表演或公开数字化执行作品。以上内容适用于纳入集体作品的作品，但这并不要求作品本身以外的集体作品必须遵守本许可的条款。如果您创建集体作品，则在征得任何许可人的通知后，您必须在切实可行的范围内，根据要求从集体作品中删除对该许可人或原始作者的任何提及。如果您创建衍生作品，则应在任何许可人的通知下，在切实可行的范围内，根据要求从衍生作品中删除对该许可人或原始作者的任何引用。根据要求从集体作品中删除对许可人或原始作者的任何引用。如果您创建衍生作品，则应在任何许可人的通知下，在切实可行的范围内，根据要求从衍生作品中删除对该许可人或原始作者的任何引用。根据要求从集体作品中删除对许可人或原始作者的任何引用。如果您创建衍生作品，则应在任何许可人的通知下，在切实可行的范围内，根据要求从衍生作品中删除对该许可人或原始作者的任何引用。

   b. 您只能在本许可，与本许可具有相同许可元素的本许可的更高版本或包含相同许可内容的Creative Commons iCommons许可下，分发，公开展示，公开表演或以数字方式进行衍生作品作为此许可证的许可证元素（例如，Attribution-NonCommercial-ShareAlike 2.0 Japan）。您必须包括以下内容的副本或统一资源标识符：您分发，公开展示，公开表演或公开数字化表演的每份衍生作品的每份副本或录音，本许可或前一句中指定的其他许可。您不得在衍生作品上提供或施加任何条款来更改或限制本许可的条款或接受者行使本协议授予的权利，并且您必须保留所有提及本许可和免责声明的声明。 。您不得散布，公开展示，公开表演，或以任何与本许可协议条款相抵触的方式，以控制其访问或使用的任何技术措施，以数字方式公开执行衍生作品。以上内容适用于纳入集体作品中的衍生作品，但这并不要求将衍生作品本身以外的集体作品置于本许可条款的约束之下。

   c. 您不得以主要旨在或针对商业利益或私人金钱补偿的方式行使以上第3节中授予您的任何权利。通过数字文件共享或其他方式将作品交换为其他受版权保护的作品，不应视为旨在或针对商业利益或私人货币补偿，只要没有与交换相关的任何货币补偿版权作品。

   d. 如果您分发，公开展示，公开表演或以数字方式表演该作品或任何衍生作品或集体作品，则您必须保留该作品的所有版权声明，并以合理的方式将原始作者的信誉给予您正在使用的媒介或手段，传达原始作者的姓名（或假名，如果适用）；作品的标题（如果提供）；在合理可行的范围内，许可人指定与作品相关联的统一资源标识符（如果有），除非此类URI并未引用该作品的版权声明或许可信息；如果是衍生作品，则应注明该作品在衍生作品中的使用（例如"原始作者的法语翻译"或"基于原始作者的原著的剧本"）。这种信用可以以任何合理的方式实施；但是，如果是衍生作品或集体作品，

   e. 为避免疑问，在作品是音乐作品的情况下：

      i. 一揽子许可下的表演版税。许可人保留单独或通过表演权利协会（例如ASCAP，BMI，SESAC）收集作品的公开表演或公共数字表演（例如网络广播）的特许权使用费的权利，如果该表演主要是针对或指导的争取商业利益或私人货币补偿。

      ii. 机械权利和法定特许权使用费。许可方保留专有权利，无论是单独还是通过音乐版权代理机构或指定的代理人（例如，Harry Fox代理机构），都可以收集您在作品中创建的任何唱片的版权费（"封面版本"），并根据所创建的强制许可进行分发由《美国版权法》（或其他司法管辖区中的等同内

容）的17 USC第115条所规定 如果您分发此类封面版本主要是出于商业利益或私人货币补偿的目的或针对的。6.网络广播权利和法定特许权使用费。为免生疑问，在作品是录音的情况下，许可方保留单独或通过表演权利协会（例如SoundExchange）收集作品的公开数字表演（例如网络广播）版税的专有权，

f. 网络广播权利和法定特许权使用费。为免生疑问，在作品是录音的情况下，许可方保留单独或通过表演权利协会（例如SoundExchange）收集作品的公开数字表演（例如网络广播）版税的专有权，遵守《美国版权法》第17条USC第114条（或其他司法管辖区中的同等规定）所规定的强制性许可，

5. 陈述，保证和免责声明

除非双方达成书面协议，否则许可方会按原样提供该工作，并且对该工作不做任何形式的明示或暗示的担保或明示，暗示，法定或其他形式的担保，包括但不限于此，保证，担保，适用于特定目的，无侵权或是否存在延迟或其他缺陷，准确性或是否存在错误，无论是否无法发现。某些司法管辖区不允许排除默示保证，因此这种排除可能不适用于您。

6. 责任限制。除适用法律要求的范围外，在任何法律理论下，对于因本许可或使用本许可证或使用本许可证而引起的任何特殊，偶发，继发性，惩罚性或特殊性损害，在任何法律理论下，许可方均不对您承担责任已被告知可能发生此类损害。

7. 终止

a. 如果您违反本许可的条款，本许可及其下授予的权利将自动终止。但是，根据本许可从您那里获得衍生作品或集体作品的个人或实体，不会终止其许可，只要这些个人或实体仍完全遵守这些许可。第1、2、5、6、7和8节将在本许可的任何终止后继续有效。

b. 根据上述条款和条件，此处授予的许可是永久性的（在作品中适用版权的期限内）。尽管如此，许可方保留在不同许可条款下发布作品或随时停止分发作品的权利；但是，任何此类选择均不得撤回本许可（或根据本许可的条款已授予或要求授予的任何其他许可），

8. 杂

a. 每次您分发或公开以数字方式执行作品或集体作品时，许可人均以与本许可下授予您的许可相同的条款和条件向接收者提供作品许可。

b. 每次您分发或公开以数字方式执行衍生作品时，许可人都会以与本许可下授予您的许可相同的条款和条件向接收者提供原始作品的许可。

c. 如果本许可的任何规定在适用法律下无效或无法执行，则不应影响本许可其余条款的有效性或可执行性，并且在本协议的各方未采取进一步行动的情况下，应将该规定修改为使此类规定有效且可执行的最小范围。

d. 除非被视为放弃或同意，否则该许可的任何条款或规定均不得视为已被放弃，也不得违反任何同意，除非该放弃或同意是由当事方书面签署的。

e. 本许可构成双方之间就此处许可的作品达成的全部协议。对于此处未指定的作品，没有任何理解，协议或陈述。许可方不受您的任何来信中可能出现的任何其他规定的约束。未经许可方和您的书面同意，不得修改本许可。

重要

知识共享不是本许可协议的任何一方，也不对与本作品有关的任何担保。对于任何损害赔偿，包括但不限于与本许可有关的任何一般性，特殊性，偶然性或后果性损害，Creative Commons对任何人或任何法律理论不承担任何责任。尽管有前两（2）句话，如果知识共享在本协议下明确表明自己是许可方，则它应具有许可方的所有权利和义务。

除向公众表明该作品是根据CCPL许可的有限目的外，未经知识共享事先书面同意，任何一方均不得使用商标“知识共享”或知识共享的任何相关商标或徽标。任何允许的使用都将符合Creative Commons当时最新的商标使用指南，该指南可能会在其网站上发布，或根据要求不时提供。

可以通过http://creativecommons.org/与知识共享联系。

F.2。麻省理工学院执照

指数

配套

工具： Diffutils-3.7
E2fsprogs： E2fsprogs-1.45.3
Eudev： Eudev-3.2.8
配置： 配置Eudev
外籍人士： 外籍人士2.2.7
预期： Expect-5.45.4
文件： File-5.37
工具： 文件5.37
Findutils： Findutils-4.6.0
工具： Findutils-4.6.0
弹性： Flex-2.6.4
Gawk： Gawk-5.0.1
工具： Gawk-5.0.1
GCC： GCC-9.2.0
工具，libstdc ++： GCC-9.2.0中的Libstdc ++
工具，通过1： GCC-9.2.0-通过1
工具，通过2：