

Linux From Scratch

Versão 13.1-systemd

Publicado 01 de setembro de 2026

Criado por Gerard Beekmans

Editor-chefe: Bruce Dubbs

Editor: Xi Ruoyao

Editor: Joe Locash

Editor: Douglas R. Reno

Linux From Scratch: Versão 13.1-systemd: Publicado 01 de setembro de 2026

por Criado por Gerard Beekmans, Editor-chefe: Bruce Dubbs, Editor: Xi Ruoyao, Editor: Joe Locash, e Editor: Douglas R. Reno
Copyright © 1999-2026 Gerard Beekmans

Direitos autorais © 1999-2026, Gerard Beekmans

Todos os direitos reservados.

Este livro é licenciado sob uma Licença da Creative Commons.

As instruções de computador tem permissão para serem extraídas a partir do livro sob a Licença do MIT.

Linux® é uma marca comercial registrada do Linus Torvalds.

Índice

Prefácio	viii
i. Apresentação	viii
ii. Audiência	ix
iii. Arquiteturas Alvo do LFS	ix
iv. Pré-requisitos	x
v. O LFS e os Padrões	x
vi. Justificativa para os Pacotes no Livro	xii
vii. Tipografia	xvii
viii. Estrutura	xix
ix. Errata e Avisos de Segurança	xix
I. Introdução	1
1. Introdução	2
1.1. Como Construir um Sistema LFS	2
1.2. O que há de novo desde o lançamento mais recente	2
1.3. Registro das Mudanças	4
1.4. Recursos	8
1.5. Ajuda	9
II. Preparando para a Construção	12
2. Preparando o Sistema Anfitrião	13
2.1. Introdução	13
2.2. Exigências do Sistema Anfitrião	13
2.3. Construindo o LFS em Estágios	16
2.4. Criando uma Nova Partição	17
2.5. Criando um Sistema de Arquivos na Partição	20
2.6. Configurando a Variável \$LFS e o Umask	21
2.7. Montando a Nova Partição	22
3. Pacotes e Remendos	24
3.1. Introdução	24
3.2. Todos os Pacotes	25
3.3. Remendos Necessários	34
4. Preparações Finais	36
4.1. Introdução	36
4.2. Criando um Layout Limitado de Diretório no Sistema de Arquivos do LFS	36
4.3. Adicionando o(a) Usante LFS	36
4.4. Configurando o Ambiente	37
4.5. A Respeito de UPCs	40
4.6. A Respeito das Suítes de Teste	41
III. Construindo o Conjunto de Ferramentas Cruzadas do LFS e Ferramentas Temporárias	42
Material Preliminar Importante	xliii
i. Introdução	xliii
ii. Observações Técnicas do Conjunto de Ferramentas	xliii
iii. Instruções Gerais de Compilação	xlix
5. Compilando um Conjunto Cruzado de Ferramentas	51
5.1. Introdução	51
5.2. Binutils-2.47 - Passagem 1	52
5.3. GCC-16.2.0 - Passagem 1	54
5.4. Cabeçalhos da API do Linux-7.1.8	57
5.5. Glibc-2.44	58

5.6. Libstdc++ oriundo de GCC-16.2.0	62
6. Compilando Cruzadamente Ferramentas Temporárias	64
6.1. Introdução	64
6.2. M4-1.4.21	65
6.3. Ncurses-6.6	66
6.4. Bash-5.3	68
6.5. Coreutils-9.11	69
6.6. Diffutils-3.12	70
6.7. File-5.48	71
6.8. Findutils-4.11.0	72
6.9. Gawk-5.4.1	73
6.10. Grep-3.12	74
6.11. Gzip-1.14	75
6.12. Make-4.4.1	76
6.13. Patch-2.8	77
6.14. Sed-4.10	78
6.15. Tar-1.35	79
6.16. Xz-5.8.3	80
6.17. Binutils-2.47 - Passagem 2	81
6.18. GCC-16.2.0 - Passagem 2	82
7. Entrando no Chroot e Construindo Ferramentas Temporárias Adicionais	84
7.1. Introdução	84
7.2. Mudando Propriedade	84
7.3. Preparando Sistemas de Arquivos Virtuais do Núcleo	84
7.4. Entrando no Ambiente Chroot	86
7.5. Criando Diretórios	86
7.6. Criando Arquivos Essenciais e Linques Simbólicos	87
7.7. Gettext-1.0	90
7.8. Bison-3.8.2	91
7.9. Perl-5.44.0	92
7.10. Zlib-1.3.2	93
7.11. mpdecimal-4.0.1	94
7.12. Python-3.14.7	95
7.13. Texinfo-7.3	96
7.14. Util-linux-2.42.2	97
7.15. Organizando e Salvando o Sistema Temporário	98
IV. Construindo o Sistema LFS	100
8. Instalando Aplicativos Básicos de Sistema	101
8.1. Introdução	101
8.2. Gerenciamento de Pacote	102
8.3. Man-pages-6.18	107
8.4. Iana-Etc-20260805	108
8.5. Glibc-2.44	109
8.6. Zlib-1.3.2	117
8.7. Bzip2-1.0.8	118
8.8. Xz-5.8.3	120
8.9. Lz4-1.10.0	122
8.10. Zstd-1.5.7	123
8.11. File-5.48	124
8.12. Readline-8.3	125

8.13. Pcre2-10.47	127
8.14. M4-1.4.21	129
8.15. Bc-7.0.3	130
8.16. Flex-2.6.4	131
8.17. Tcl-8.6.18	132
8.18. Expect-5.45.4	134
8.19. DejaGNU-1.6.3	136
8.20. Ninja-1.13.2	137
8.21. Pkgconf-3.0.5	138
8.22. Binutils-2.47	139
8.23. GMP-6.3.0	142
8.24. MPFR-4.2.2	144
8.25. MPC-1.4.1	145
8.26. Attr-2.6.0	146
8.27. Acl-2.4.0	147
8.28. Libcap-2.78	148
8.29. Libxcrypt-4.5.2	149
8.30. Shadow-4.20.2	151
8.31. Gawk-5.4.1	155
8.32. GCC-16.2.0	156
8.33. Ncurses-6.6	162
8.34. Sed-4.10	165
8.35. Psmisc-23.7	166
8.36. Gettext-1.0	167
8.37. Bison-3.8.2	169
8.38. Grep-3.12	170
8.39. Bash-5.3	171
8.40. Libtool-2.6.2	173
8.41. GDBM-1.26	174
8.42. Gperf-3.3	175
8.43. Expat-2.8.3	176
8.44. Inetutils-2.8	177
8.45. Less-704	179
8.46. Perl-5.44.0	180
8.47. Autoconf-2.73	183
8.48. Automake-1.18.1	184
8.49. OpenSSL-4.0.1	185
8.50. Libelf originário do Elfutils-0.195	187
8.51. Libffi-3.8.0	188
8.52. Sqlite-3.53.0	189
8.53. mpdecimal-4.0.1	191
8.54. Python-3.14.7	192
8.55. Flit-Core-4.0.2	195
8.56. Packaging-26.3	196
8.57. Wheel-0.48.0	197
8.58. Setuptools-84.0.0	198
8.59. Meson-1.12.0	199
8.60. Kmod-34.2	200
8.61. Coreutils-9.11	201
8.62. Diffutils-3.12	206

8.63. Findutils-4.11.0	207
8.64. Groff-1.24.1	208
8.65. GRUB-2.14	211
8.66. Gzip-1.14	215
8.67. IPRoute2-7.1.0	216
8.68. Kbd-2.10.0	218
8.69. Libpipeline-1.5.8	221
8.70. Make-4.4.1	222
8.71. Patch-2.8	223
8.72. Tar-1.35	224
8.73. Texinfo-7.3	225
8.74. Vim-9.2.1025	227
8.75. MarkupSafe-3.0.3	230
8.76. Jinja2-3.1.6	231
8.77. Systemd-261.2	232
8.78. D-Bus-1.16.2	238
8.79. Man-DB-2.13.1	240
8.80. Procps-ng-4.0.7	243
8.81. Util-linux-2.42.2	245
8.82. E2fsprogs-1.47.4	250
8.83. Acerca dos Símbolos de Depuração	253
8.84. Despojando	253
8.85. Organizando	255
9. Configuração do Sistema	256
9.1. Introdução	256
9.2. Configuração Geral da Rede de Comunicação	256
9.3. Visão Geral do Manuseio de Dispositivo e de Módulo	260
9.4. Gerenciando Dispositivos	264
9.5. Configurando o Relógio do Sistema	264
9.6. Configurando o Console do Linux	265
9.7. Configurando a Localidade do Sistema	267
9.8. Criando o Arquivo /etc/inputrc	269
9.9. Criando o Arquivo /etc/shells	270
9.10. Uso e Configuração do Systemd	271
10. Tornando o Sistema LFS Inicializável	275
10.1. Introdução	275
10.2. Criando o Arquivo /etc/fstab	275
10.3. Linux-7.1.8	277
10.4. Usando o GRUB para Configurar o Processo de Inicialização	285
11. O Fim	290
11.1. O Fim	290
11.2. Seja Contado(a)	290
11.3. Reinicializando o Sistema	290
11.4. Recursos Adicionais	291
11.5. Começando Depois do LFS	292
V. Anexos	295
A. Siglas e Termos	296
B. Reconhecimentos	298
C. Dependências	301
D. Licenças do LFS	321

D.1. Licença da Creative Commons	321
D.2. A Licença do MIT	325
Índice Remissivo	326

Prefácio

Apresentação

Minha jornada para aprender e entender melhor o Linux começou em meados de 1998. Eu havia acabado de instalar minha primeira distribuição do Linux e rapidamente fiquei intrigado com todo o conceito e filosofia por trás do Linux.

Existem sempre muitas maneiras de realizar uma tarefa. O mesmo pode ser dito a respeito das distribuições do Linux. Um grande número existiu ao longo dos anos. Algumas ainda existem; algumas se transformaram em algo mais; e ainda outras foram relegadas às nossas memórias. Todas elas fazem coisas diferentemente para se adequarem às necessidades da audiência alvo delas. Devido a existirem muitíssimas maneiras de realizar o mesmo objetivo final, eu comecei a perceber que não tinha que estar limitado por qualquer uma implementação. Antes de descobrir o Linux, nós simplesmente lidávamos com problemas em outros Sistemas Operacionais como se você não tivesse escolha. A coisa era o que era, não importando se você gostasse ou não. Com o Linux, o conceito de escolha começou a emergir. Se você não gostou de alguma coisa, você seria livre, até encorajado(a), a mudá-la.

Eu tentei várias distribuições e não consegui me decidir por nenhuma. Elas eram grandes sistemas em seu próprio direito. Não era mais uma questão de certo ou errado. Tinha se tornado em uma questão de gosto pessoal. Com todas aquelas escolhas disponíveis, tornou-se aparente que não haveria um sistema que fosse perfeito para mim. Então eu me propus a criar meu próprio sistema Linux, que estaria totalmente em conformidade com minhas preferências pessoais.

Para verdadeiramente torná-lo meu próprio sistema, eu resolvi compilar tudo a partir do código fonte, em vez de usar pacotes pré compilados de binário. Esse sistema Linux “perfeito” teria a força de vários sistemas sem suas fraquezas visíveis. A princípio, a ideia era bastante amedrontadora. Eu me mantive comprometido com a ideia de que tal sistema poderia ser construído.

Depois de lidar com problemas, tais como dependências circulares e erros em tempo de compilação, eu finalmente construí um sistema Linux feito sob encomenda. Era totalmente operacional e perfeitamente usável, como quaisquer dos outros sistemas Linux disponíveis na época. Porém, era minha própria criação. Foi muito gratificante ter montado tal sistema eu mesmo. A única coisa melhor teria sido criar cada pedaço de software eu mesmo. Essa foi a melhor coisa que se seguiu.

Conforme eu compartilhei meus objetivos e experiências com outros(as) membros(as) da comunidade Linux, tornou-se aparente que havia um interesse firme nessas ideias. Logo tornou-se claro que tais sistemas Linux feitos sob encomenda serviam não somente para satisfazer as exigências específicas de usantes, mas também serve como uma oportunidade ideal de aprendizado para programadores(as) e administradores(as) de sistema elevarem as (existentes) habilidades deles(as) com o Linux. Como resultado desse interesse amplo, o *Projeto Linux From Scratch* nasceu.

Este livro Linux From Scratch é o núcleo em torno desse projeto. Ele provê a base e as instruções necessárias para você projetar e construir teu próprio sistema. Ao tempo em que este livro fornece um modelo que resultará em um sistema que funciona corretamente, você é livre para alterar as instruções conforme tua preferência, o que é, em parte, uma importante parte deste projeto. Você permanece no controle; nós só damos uma mão para ajudá-lo(a) a começar na tua própria jornada.

Eu sinceramente espero que você tenha um ótimo tempo trabalhando em teu próprio sistema Linux From Scratch e que aproveite os numerosos benefícios de ter um sistema que é verdadeiramente teu.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Audiência

Existem muitas razões pelas quais você desejaria ler este livro. Uma das questões que muitas pessoas levantam é “por que ir ao longo de toda a dificuldade de construir manualmente um sistema Linux desde o zero quando você pode simplesmente baixar e instalar um existente?”

Uma importante razão para a existência desse projeto é para te ajudar a aprender como um sistema Linux funciona de dentro para fora. Construir um sistema LFS ajuda a demonstrar o que torna o Linux de interesse e como as coisas funcionam juntas e dependem umas das outras. Uma das melhores coisas que essa experiência de aprendizado pode fornecer é a habilidade para personalizar um sistema Linux para se ajustar às suas próprias necessidades únicas.

Outro benefício chave do LFS é o de que ele te dá controle do sistema sem confiar na implementação Linux de ninguém. Com o LFS, você está no banco do motorista. *Você* dita cada aspecto do teu sistema.

O LFS te permite criar sistemas Linux muito compactos. Com outras distribuições, você frequentemente é forçado(a) a instalar grande número de programas, os quais nem usa, nem entende. Esses programas desperdiçam recursos. Você possivelmente argumente que, com as unidades rígidas e CPUs de hoje, recursos desperdiçados não mais são uma consideração. Às vezes, entretanto, você ainda está restrito(a) pelo tamanho do sistema, se nenhuma outra coisa. Pense a respeito de CDs inicializáveis, mídias USB e sistemas embarcados. Essas são áreas onde o LFS pode ser benéfico.

Outra vantagem de um sistema Linux feito sob medida é a da segurança. Ao compilar o sistema inteiro desde o zero, você está empoderado(a) para auditar tudo e aplicar todos os remendos de segurança que queira. Você não tem que aguardar que outra pessoa compile os pacotes binários que consertam uma brecha de segurança. A menos que você examine o remendo e o implemente você mesma(o), você não tem garantias de que o novo pacote binário foi construído corretamente e adequadamente conserta o problema.

O objetivo do Linux From Scratch é o de construir um sistema em nível de fundação completo e usável. Se você não deseja construir teu próprio sistema Linux desde o zero, você possivelmente nunca se beneficie das informações neste livro.

Existem muito mais boas razões para construir teu próprio sistema LFS para listá-las todas aqui. No final, educação é, de longe, a mais importante razão. Conforme continue tua experiência LFS, você descobrirá o poder que informação e conhecimento podem trazer.

Arquiteturas Alvo do LFS

The primary target architectures of LFS are the AMD/Intel x86 (32-bit) and x86_64 (64-bit) CPUs. On the other hand, the instructions in this book are also known to work, with some modifications, with the Power PC and ARM CPUs. To build a system that utilizes one of these alternative CPUs, the main prerequisite, in addition to those on the next page, is an existing Linux system such as an earlier LFS installation, Ubuntu, Red Hat/Fedora, SuSE, or some other distribution that targets that architecture. (Note that a 32-bit distribution can be installed and used as a host system on a 64-bit AMD/Intel computer.)

O ganho oriundo da construção em um sistema de 64 bits comparado a um sistema de 32 bits é mínimo. Por exemplo, em uma construção de teste do LFS-9.1 em um sistema baseado na CPU Core i7-4790, usando quatro núcleos, as seguintes estatísticas foram verificadas:

Arquitetura	Tempo de Construção	Tamanho de Construção
32 bits	239,9 minutos	3,6 GB
64 bits	233,2 minutos	4,4 GB

Como você pode ver, no mesmo hardware, a construção de 64 bits é somente 3% mais rápida (e 22% maior) que a construção de 32 bits. Se você planeja usar o LFS como um servidor LAMP, ou como um firewall, uma CPU de 32 bits possivelmente seja boa o suficiente. Por outro lado, vários pacotes no BLFS atualmente precisam de mais que 4 GB de RAM para serem construídos e (ou) para executarem; se você planeja usar o LFS como um desktop, os(as) autores(as) do LFS recomendam construir um sistema de 64 bits.

The default 64-bit build that results from LFS is a “pure” 64-bit system. That is, it supports 64-bit executables only. Building a “multi-lib” system requires compiling many applications twice, once for a 32-bit system and once for a 64-bit system. This is not directly supported in LFS because it would interfere with the educational objective of providing the minimal instructions needed for a basic Linux system. Some of the LFS/BLFS editors maintain a multilib fork of LFS, accessible at <https://www.linuxfromscratch.org/~thomas/multilib/index.html>. But that's an advanced topic.

Pré-requisitos

Construir um sistema LFS não é uma tarefa simples. Exige um certo nível de conhecimento existente da administração de sistemas Unix para a finalidade de resolver problemas e corretamente executar os comandos listados. Em particular, como um mínimo absoluto, você já deveria saber como usar a linha de comando (shell) para copiar ou mover arquivos e diretórios, listar diretórios e conteúdo de arquivos, e mudar o diretório atual. Também é esperado que você saiba como usar e instalar software Linux.

Por causa do livro LFS assumir *pelo menos* esse nível básico de habilidades, os vários fóruns de suporte do LFS não são adequados para te fornecer muita assistência nessas áreas. Você vai achar que as suas perguntas relacionadas a tal conhecimento básico não serão respondidas (ou serão simplesmente remetidas à lista de pré-leitura essencial do LFS).

Antes de construir um sistema LFS, nós insistimos que você leia estes artigos:

- Software-Building-HOWTO <http://www.tldp.org/HOWTO/Software-Building-HOWTO.html>
Esse é um guia compreensivo para construir e instalar pacotes de software Unix “genéricos” sob o Linux. Apesar de que foi escrito há algum tempo, ainda fornece um bom resumo das técnicas básicas usadas para construir e instalar software.
- Guia do(a) Iniciante para Instalar a partir do Fonte <http://moi.vonos.net/linux/beginners-installing-from-source/>
Esse guia fornece um bom sumário das habilidades básicas e de técnicas necessárias para construir software a partir do código fonte.

O LFS e os Padrões

A estrutura do LFS segue os padrões do Linux tão rigorosamente quanto possível. Os padrões primários são:

- *POSIX.1-2008*.
- *Filesystem Hierarchy Standard (FHS) Versão 3.0*
- *Linux Standard Base (LSB) Versão 5.0 (2015)*

O LSB tem quatro especificações: Core, Desktop, Languages e Imaging. Algumas partes das especificações Core e Desktop são específicas de arquitetura. Existem também duas especificações experimentais: Gtk3 e Graphics. O LFS tenta obedecer às especificações da LSB para as arquiteturas IA32 (x86 de 32 bits) ou AMD64 (x86_64) discutidas na sessão anterior.



Nota

Muitas pessoas não concordam com essas exigências. O principal propósito do LSB é o de garantir que software proprietário consiga ser instalado e execute em um sistema compatível. Dado que o LFS é baseado no fonte, o(a) usante tem total controle sobre quais pacotes são desejados; você possivelmente escolha não instalar alguns pacotes que são especificados pelo LSB.

Embora seja possível criar um sistema completo que passará nos testes de certificação da LSB “desde o zero”, isso não pode ser feito sem muitos pacotes adicionais que estão além do escopo do livro LFS. Instruções de instalação para alguns desses pacotes adicionais podem ser encontradas no BLFS.

Pacotes fornecidos pelo LFS necessários para satisfazer as Exigências do LSB

<i>Núcleo do LSB:</i>	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, M4, Man-DB, Procps, Psmisc, Sed, Shadow, Systemd, Tar, Util-linux, Zlib
<i>Área de trabalho do LSB:</i>	Nenhum
<i>Linguagens da LSB:</i>	Perl
<i>Imagem no LSB:</i>	Nenhum
<i>LSB Gtk3 e Gráficos LSB (Uso Experimental):</i>	Nenhum

Pacotes fornecidos pelo BLFS necessários para satisfazer as Exigências do LSB

<i>Núcleo do LSB:</i>	At, Batch (uma parte de At), Arquivos do BLFS de Iniciação do Bash, Cpio, Ed, Fcron, LSB-Tools, NSPR, NSS, Linux-PAM, Pax, Sendmail (ou Postfix ou Exim), Time
<i>Área de trabalho do LSB:</i>	Aalsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GLU, Icon-naming-utils, Libjpeg-turbo, Libxml2, Mesa, Pango, Xdg-utils, Xorg
<i>Linguagens da LSB:</i>	Libxml2 e Libxslt
<i>Imagem no LSB:</i>	CUPS, Cups-filters, Ghostscript e SANE
<i>LSB Gtk3 e Gráficos LSB (Uso Experimental):</i>	GTK+3

Componentes não fornecidos ou opcionalmente fornecidos pelo LFS ou pelo BLFS necessários para satisfazer as Exigências da LSB

<i>Núcleo do LSB:</i>	install_initd , libcrypt.so.1 (pode ser fornecido com instruções opcionais para o pacote Libxcrypt do LFS), libncurses.so.5 (pode ser fornecido com instruções opcionais para o pacote Ncurses do LFS), libncursesw.so.5 (mas libncursesw.so.6 é fornecido pelo pacote Ncurses do LFS)
<i>Área de trabalho do LSB:</i>	libgdk-x11-2.0.so (mas libgdk-3.so é fornecido pelo pacote GTK+3 do BLFS), libgtk-x11-2.0.so (mas libgtk-3.so e libgtk-4.so são fornecidos pelos pacotes GTK+3 e GTK-4 do BLFS), libpng12.so (mas libpng16.so é fornecido pelo pacote Libpng do BLFS), libqt*.so.4 (mas libqt6*.so.6 são fornecidos pelo pacote Qt6 do BLFS), libtiff.so.4 (mas libtiff.so.6 é fornecido pelo pacote Libtiff do BLFS)
<i>Linguagens da LSB:</i>	/usr/bin/python (LSB exige Python2, mas LFS e BLFS fornecem somente Python3)
<i>Imagem no LSB:</i>	Nenhum
<i>LSB Gtk3 e Gráficos LSB (Uso Experimental):</i>	libpng15.so (mas libpng16.so é fornecido pelo pacote Libpng do BLFS)

Justificativa para os Pacotes no Livro

O objetivo do LFS é o de construir um sistema em nível de fundação completo e usável—incluindo todos os pacotes necessários para replicar a ele mesmo—e fornecer uma base relativamente mínima a partir da qual personalizar um sistema mais completo baseado nas escolhas do(a) usante. Isso não significa que o LFS é o menor sistema possível. Vários pacotes importantes estão inclusos que não são, falando estritamente, exigidos. A lista abaixo documenta ao motivos para cada pacote no livro ter sido incluído.

- Acl

Esse pacote contém utilitários para administrar Listas de Controle de Acesso, as quais são usadas para definir direitos de acesso discricionariamente finamente refinados para arquivos e para diretórios.
- Attr

Esse pacote contém aplicativos para gerenciar atributos estendidos sobre objetos do sistema de arquivos.
- Autoconf

Esse pacote fornece aplicativos para produzir scripts de shell que podem configurar automaticamente o código fonte a partir de um modelo do(a) desenvolvedor(a). Frequentemente é necessário para reconstruir um pacote depois que o procedimento de construção tenha sido atualizado.
- Automake

Esse pacote contém aplicativos para gerar arquivos Make a partir de um modelo. Frequentemente é necessário para reconstruir um pacote depois que o procedimento de construção tenha sido atualizado.
- Bash

Esse pacote satisfaz uma exigência central do LSB para fornecer uma interface Bourne Shell para o sistema. Foi escolhido em vez de outros pacotes de shell por causa do uso comum e capacidades extensas dele.
- Bc

Esse pacote fornece uma linguagem de processamento numérico com precisão arbitrária. Satisfaz uma exigência para construir o núcleo do Linux.
- Binutils

Esse pacote fornece um vinculador, um montador e outras ferramentas para manusear arquivos objeto. Os aplicativos nesse pacote são necessários para compilar a maioria dos pacotes em um sistema LFS.
- Bison

Esse pacote contém a versão GNU do yacc (Yet Another Compiler Compiler) necessário para construir vários dos aplicativos do LFS.
- Bzip2

Esse pacote contém aplicativos para comprimir e descomprimir arquivos. É exigido para descomprimir muitos pacotes do LFS.
- Coreutils

Esse pacote contém um número de aplicativos essenciais para visualizar e manipular arquivos e diretórios. Esses aplicativos são necessários para o gerenciamento de arquivos por linha de comando e são necessários para os procedimentos de instalação de cada pacote no LFS.
- D-Bus

Esse pacote contém aplicativos para implementar um sistema de barramento de mensagem, uma maneira simples para aplicativos conversarem com um outro.

- DejaGNU

Esse pacote fornece uma estrutura para testar outros aplicativos.

- Diffutils

Esse pacote contém aplicativos que mostram as diferenças entre arquivos ou diretórios. Esses aplicativos podem ser usados para criar remendos e também são usados em muitos procedimentos de construção dos pacotes.

- E2fsprogs

Esse pacote fornece utilitários para manusear os sistemas de arquivos ext2, ext3 e ext4. Esses são os sistemas de arquivos mais comuns e amplamente testados que o Linux suporta.

- Expat

Esse pacote produz uma biblioteca de análise relativamente pequena de XML. É exigida pelo módulo do Perl `XML::Parser`.

- Expect

Esse pacote contém um aplicativo para realizar diálogos com scripts com outros aplicativos interativos. É comumente usado para testar outros pacotes.

- File

Esse pacote contém um utilitário para determinar o tipo de um dado arquivo ou arquivos. Uns poucos pacotes precisam dele nos scripts de construção deles.

- Findutils

Esse pacote fornece aplicativos para encontrar arquivos em um sistema de arquivos. É usado em muitos scripts de construção dos pacotes.

- Flex

Esse pacote contém um utilitário para gerar aplicativos que reconhecem padrões em texto. É a versão GNU do aplicativo `lex` (lexical analyzer). É exigido para construir vários pacotes do LFS.

- Gawk

Esse pacote fornece aplicativos para manipular arquivos de texto. É a versão GNU do `awk` (Aho-Weinberg-Kernighan). É usado em muitos outros scripts de construção dos pacotes.

- GCC

Esse é o Gnu Compiler Collection. Contém os compiladores C e C++ assim como vários outros não construídos pelo LFS.

- GDBM

Esse pacote contém a biblioteca GNU Database Manager. É usado por um outro pacote do LFS, `Man-DB`.

- Gettext

Esse pacote fornece utilitários e bibliotecas para a internacionalização e localização de muitos pacotes.

- Glibc

Esse pacote contém a biblioteca C principal. Aplicativos Linux não executarão sem ela.

- GMP

Esse pacote fornece bibliotecas matemáticas que fornecem funções úteis para aritmética de precisão arbitrária. É necessário para construir o GCC.

- Gperf

Esse pacote produz um aplicativo que gera uma função perfeita de resumo a partir de um conjunto de chaves. É exigido pelo Systemd.

- Grep

Esse pacote contém aplicativos para pesquisar ao longo de arquivos. Esses aplicativos são usados pela maioria dos scripts de construção dos pacotes.

- Groff

Esse pacote contribui com aplicativos para processar e formatar texto. Uma função importante desses aplicativos é a de formatar páginas de manual.

- GRUB

Esse pacote é o Grand Unified Boot Loader. É o mais flexível dos vários carregadores de inicialização disponíveis.

- Gzip

Esse pacote contém aplicativos para comprimir e descomprimir arquivos. É necessário para descomprimir muitos pacotes no LFS.

- Iana-etc

Esse pacote fornece dados para serviços e protocolos de rede de comunicação. É necessário para habilitar recursos adequados da rede de comunicação.

- Inetutils

Esse pacote fornece aplicativos para administração básica da rede de comunicação.

- IProute2

Esse pacote contém aplicativos para redes de comunicação IPv4 e IPv6 básicas e avançadas. Ele foi escolhido em vez dos outros pacotes comuns de ferramentas de rede de comunicação (net-tools) pelos recursos de IPv6 dele.

- Jinja2

Esse pacote é um módulo do Python para modelagem de texto. É exigido para construir o Systemd.

- Kbd

Esse pacote produz arquivos de tabelas chave, utilitários de teclado para teclados que não sejam estadunidenses e um número de fontes de console.

- Kmod

Esse pacote fornece aplicativos necessários para administrar os módulos do núcleo Linux.

- Less

Esse pacote contém um visualizador de arquivo de texto muito bom que permite rolar para cima ou para baixo quando se visualiza um arquivo. Muitos pacotes o usam para paginar a saída gerada.

- Libcap

Esse pacote implementa as interfaces do espaço de usuário para os recursos POSIX 1003.1 e disponíveis nos núcleos Linux.

- Libelf

O projeto elfutils fornece bibliotecas e ferramentas para arquivos ELF e dados DWARF. A maior parte dos utilitários nesse pacote está disponível em outros pacotes, porém a biblioteca é necessária para construir o núcleo Linux usando a configuração padrão (e mais eficiente).

- Libffi

Esse pacote implementa uma interface de programação portátil, de alto nível, para várias convenções de chamada. Alguns aplicativos possivelmente não saibam, ao tempo da compilação, quais argumentos são para serem passados para uma função. Por exemplo, um interpretador possivelmente possa ser informado, ao tempo de execução, acerca do número e dos tipos de argumentos usados para chamar uma dada função. Libffi pode ser usada em tais aplicativos para fornecer uma ponte a partir do aplicativo interpretador para o código compilado.

- Libpipeline

O pacote Libpipeline fornece uma biblioteca para manipular pipelines de subprocessos de uma maneira flexível e conveniente. Ele é exigido pelo pacote Man-DB.

- Libtool

Esse pacote contém o script GNU de suporte a bibliotecas genéricas. Ele esconde a complexidade do uso de bibliotecas compartilhadas em uma interface consistente e portátil. Ele é necessário para as suítes de testes em outros pacotes do LFS.

- Libxcrypt

Esse pacote fornece a biblioteca `libcrypt` necessária para vários pacotes (notavelmente, Shadow) para resumir senhas. Ela substitui a implementação obsoleta `libcrypt` na Glibc.

- Núcleo Linux

Esse pacote é o Sistema Operacional. Ele é o Linux no ambiente GNU/Linux.

- M4

Esse pacote fornece um processador geral de macro de texto, útil como uma ferramenta de construção para outros aplicativos.

- Make

Esse pacote contém um aplicativo para direcionar a construção de pacotes. Ele é exigido por quase todos os pacotes no LFS.

- MarkupSafe

Esse pacote é um módulo do Python para processar sequências de caracteres em HTML/XHTML/XML com segurança. Jinja2 exige esse pacote.

- Man-DB

Esse pacote contém aplicativos para encontrar e visualizar páginas de manual. Ele foi escolhido em vez do pacote `man` por causa dos recursos superiores de internacionalização dele. Ele fornece o aplicativo `man`.

- Páginas-Manual

Esse pacote fornece o conteúdo atual das páginas de manual básicas do Linux.

- Meson

Esse pacote fornece uma ferramenta de software para automatizar a construção de software. O objetivo principal do Meson é o de minimizar a quantidade de tempo que desenvolvedores(as) de software precisam investir configurando um sistema de construção. Ele é exigido para construir o Systemd, bem como muitos pacotes do BLFS.

- MPC

Esse pacote fornece funções aritméticas para números complexos. Ele é exigido pelo GCC.

- Mpdecimal

Esse pacote fornece funções aritméticas para números flutuantes decimais. Ele é exigido pelo Python.

- MPFR

Esse pacote contém funções para aritmética de precisão múltipla. Ele é exigido pelo GCC.

- Ninja

Esse pacote equipa um sistema pequeno de construção com um foco em velocidade. Ele é projetado para ter os arquivos de entrada gerada dele gerados por um sistema de construção de nível mais alto e para executar construções o mais rápido possível. Esse pacote é exigido pelo Meson.

- Ncurses

Esse pacote contém bibliotecas para o manuseio, independente de terminal, de telas de caractere.

Frequentemente é usado para fornecer controle de cursor para um sistema com menus. Ele é necessitado por um número de pacotes no LFS.

- Openssl

Esse pacote fornece ferramentas e bibliotecas de gerenciamento relacionadas à criptografia. Essas fornecem funções criptográficas para outros pacotes, incluindo o núcleo Linux.

- Patch

Esse pacote contém um aplicativo para modificar ou criar arquivos aplicando um arquivo de *remendo* tipicamente criado pelo aplicativo diff. Ele é necessitado pelo procedimento de construção para vários pacotes do LFS.

- Pcre2

Esse pacote fornece um conjunto de funções que implementam correspondência de padrões de expressões regulares usando a mesma sintaxe e semântica do Perl 5.

- Perl

Esse pacote é um interpretador para a linguagem de tempo de execução PERL. Ele é necessário para a instalação e suítes de teste de vários pacotes do LFS.

- Pkgconf

Esse pacote contém um aplicativo que ajuda a configurar sinalizadores de compilador e de vinculador para bibliotecas de desenvolvimento. O aplicativo pode ser usado como um substituto imediato do **pkg-config**, que é necessário para o sistema de construção de muitos pacotes. Ele é mantido mais ativamente e um pouco mais rápido que o pacote Pkg-config original.

- Procps-NG

Esse pacote contém aplicativos para monitorar processos. Esses aplicativos são úteis para administração de sistema e também são usados pelos scripts de inicialização do LFS.

- Psmisc

Esse pacote produz aplicativos para mostrar informações acerca de processos em execução. Esses aplicativos são úteis para administração de sistema.

- Python 3

Esse pacote fornece uma linguagem interpretada que tem uma filosofia de projeto que enfatiza a legibilidade de código.

- Readline

Esse pacote é um conjunto de bibliotecas que oferecem recursos de edição e de histórico de linha de comando. Ele é usado pelo Bash.

- Sed

Esse pacote permite edição de texto sem abri-lo em um editor de texto. Ele também é necessitado por muitos scripts de configuração dos pacotes do LFS.
- Shadow

Esse pacote contém aplicativos para manusear senhas seguramente.
- Sqlite

Esse pacote fornece um mecanismo de base de dados SQL transacional, sem servidor e com configuração zero.
- Systemd

Esse pacote fornece um programa init e vários recursos adicionais de inicialização e de controle de sistema como uma alternativa ao SysVinit. Ele é usado por muitas distribuições do Linux.
- Tar

Esse pacote fornece recursos de arquivamento e de extração de virtualmente todos os pacotes usados no LFS.
- Tcl

Esse pacote contém a Tool Command Language usada em muitas suítes de teste.
- Texinfo

Esse pacote fornece aplicativos para ler, escrever e converter páginas info. Ele é usado nos procedimentos de instalação de muitos pacotes do LFS.
- Util-linux

Esse pacote contém aplicativos utilitários diversos. Entre eles estão utilitários para manusear sistemas de arquivos, consoles, partições e mensagens.
- Vim

Esse pacote fornece um editor. Ele foi escolhido por causa da compatibilidade dele com o clássico editor vi e o número gigante de recursos poderosos dele. Um editor é uma escolha muito pessoal para muitos(as) usantes. Qualquer outro editor pode ser substituído, se você desejar.
- Wheel

Esse pacote fornece um módulo do Python que é a implementação de referência do padrão de empacotamento roda do Python.
- XZ Utils

Esse pacote contém aplicativos para comprimir e descomprimir arquivos. Ele fornece a maior compressão geralmente disponível e é útil para descomprimir pacotes no formato XZ ou LZMA.
- Zlib

Esse pacote contém rotinas de compressão e de descompressão usadas por alguns aplicativos.
- Zstd

Esse pacote fornece rotinas de compressão e de descompressão usadas por alguns aplicativos. Ele fornece taxas altas de compressão e um intervalo muito amplo de intercâmbios entre compressão / velocidade.

Tipografia

Para tornar as coisas mais fáceis de serem seguidas, existem umas poucas convenções tipográficas usadas ao longo deste livro. Esta seção contém alguns exemplos do formato tipográfico encontrado ao longo do Linux From Scratch.

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

Essa forma de texto é projetada para ser digitada exatamente como vista, a menos que seja dito o contrário no texto que a envolve. Também é usada nas seções de explicação para identificar quais dos comandos estão sendo referenciados.

Em alguns casos, uma linha lógica é estendida em duas ou mais linhas físicas com uma barra invertida no final da linha.

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

Observe que a barra invertida precisa ser seguida por uma quebra de linha imediata. Outros caracteres de espaço em branco, como caracteres de espaços ou de tabulação criarão resultados incorretos.

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

Essa forma de texto (de largura fixa) mostra a saída gerada em tela, geralmente como o resultado de comandos emitidos. Se você estiver lendo o livro no formato HTML (em vez do PDF), o texto deveria estar azul.

O texto de largura fixa também é usado para mostrar nomes de arquivos, tais como `/etc/ld.so.conf`.



Nota

Por favor, configure o seu navegador para exibir texto de largura fixa com uma boa fonte mono espaçada, com a qual você possa distinguir claramente os glifos de `l11` ou `00`.

Ênfase

Essa forma de texto é usada para vários propósitos no livro. O propósito principal dela é o de enfatizar pontos ou itens importantes.

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

Esse formato é usado para hiperlinks tanto dentro da comunidade do LFS, quanto para páginas externas. Inclui HOWTOs, locais de download e páginas da Internet.

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

Esse formato é usado quando da criação de arquivos de configuração. O primeiro comando diz ao sistema para criar o arquivo `$LFS/etc/group` a partir do que seja digitado nas linhas seguintes até que a sequência “End Of File” (EOF) seja encontrada. Portanto, essa seção inteira geralmente é digitada como é vista.

<TEXTO SUBSTITUÍDO>

Esse formato é usado para encapsular texto que não é para ser digitado como visto ou para operações de “copiar-e-colar”.

[TEXTO OPCIONAL]

Esse formato é usado para encapsular texto que é opcional.

passwd(5)

Esse formato é usado para referir-se a uma página de manual (man) específica. O número dentro dos parênteses indica uma seção específica dentro dos manuais. Por exemplo, o **passwd** tem duas páginas de manual. Conforme as instruções de instalação do LFS, essas duas páginas de manual estarão localizadas em `/usr/share/man/man1/passwd.1` e `/usr/share/man/man5/passwd.5`. Quando o livro usar *passwd(5)* ele está se referindo especificamente a `/usr/share/man/man5/passwd.5`. **man passwd** imprimirá a primeira página de manual que achar que corresponde a “passwd”, a qual será `/usr/share/man/man1/passwd.1`. Para esse exemplo, você precisará executar **man 5 passwd**

para a finalidade de ler a página sendo especificada. Observe que a maioria das páginas de manual não têm nomes duplicados de páginas em diferentes seções. Portanto, **man <nome do programa>** geralmente é suficiente. No livro LFS essas referências para páginas de manual também são hiper links, de modo que clicar-se em tal referência abrirá a página de manual renderizada em HTML a partir das *páginas de manual do Arch Linux*.

Estrutura

Este livro é dividido nas seguintes partes.

Parte I – Introdução

A Parte I explica umas poucas observações importantes a respeito do como proceder com a instalação do LFS. Essa seção também fornece metainformação a respeito do livro.

Parte II – Preparando para a Construção

A Parte II descreve como se preparar para o processo de construção—criando uma partição, baixando os pacotes e compilando as ferramentas temporárias.

Parte III – Construindo o Conjunto de Ferramentas Cruzadas e Ferramentas Temporárias do LFS

A Parte III fornece instruções para a construção das ferramentas necessárias para construir o sistema LFS final.

Parte IV - Construindo o Sistema LFS

A Parte IV guia o(a) leitor(a) ao longo da construção do sistema LFS—compilando e instalando todos os pacotes, um por um, configurando os scripts de inicialização e instalando o núcleo. O sistema Linux resultante é a fundação sobre a qual outros aplicativos podem ser construídos para expandir o sistema conforme desejado. No final deste livro, existe uma referência fácil de usar listando todos os aplicativos, bibliotecas e arquivos importantes que tenham sido instalados.

Parte V - Anexos

A Parte V fornece informação acerca do próprio livro incluindo acrônimos e termos, reconhecimentos, dependências de pacotes, uma listagem dos scripts de inicialização do LFS, licenças para a distribuição do livro e um índice compreensível dos pacotes, aplicativos, bibliotecas e scripts.

Errata e Avisos de Segurança

O software usado para criar um sistema LFS constantemente está sendo atualizado e melhorado. Avisos de segurança e correções de defeitos possivelmente se tornem disponíveis depois que o livro LFS tenha sido lançado. Para verificar se versões dos pacotes ou as instruções neste lançamento do LFS necessitam de quaisquer modificações—para reparar vulnerabilidades de segurança ou para corrigir outros defeitos—por favor visite <https://www.linuxfromscratch.org/lfs/errata/13.1-systemd/> antes de continuar com a sua construção. Você deveria observar quaisquer mudanças mostradas e aplicá-las às seções relevantes do livro conforme construa o sistema LFS.

Adicionalmente, os(as) editores(as) do Linux From Scratch mantém uma lista das vulnerabilidades de segurança descobertas *depois que* um livro tenha sido lançado. Para ler a lista, por favor visite <https://www.linuxfromscratch.org/advisories/> antes de proceder com sua construção. Você deveria aplicar as mudanças sugeridas pelos avisos às seções relevantes do livro conforme construa o sistema LFS. E, se usará o sistema LFS como um desktop real ou um sistema servidor, [então] você deveria continuar a consultar os avisos e a corrigir quaisquer vulnerabilidades de segurança, mesmo quando o sistema LFS tenha sido completamente construído.

Parte I. Introdução

Capítulo 1. Introdução

1.1. Como Construir um Sistema LFS

O sistema LFS será construído usando uma distribuição Linux já instalada (tal como Debian; o OpenMandriva; Fedora ou openSUSE). Esse sistema Linux existente (o anfitrião) será usado como um ponto de partida para fornecer os aplicativos necessários, incluindo um compilador, um vinculador e um shell para construir o novo sistema. Selecione a opção “desenvolvimento” durante a instalação da distribuição para incluir essas ferramentas.



Nota

Existem muitas maneiras de se instalar uma distribuição Linux e os padrões geralmente não são ideais para construir um sistema LFS. Para sugestões a respeito de configurar uma distribuição comercial, veja-se: <https://www.linuxfromscratch.org/hints/downloads/files/partitioning-for-lfs.txt>.

Como uma alternativa a instalar uma distribuição separada em sua máquina, você possivelmente deseje usar um LiveCD de uma distribuição comercial.



Nota

A distribuição anfitriã precisa satisfazer Exigências do Sistema Anfitrião.

O Capítulo 2 deste livro descreve como criar uma nova partição Linux nativa e sistema de arquivos, onde o novo sistema LFS será compilado e instalado. O Capítulo 3 explica quais pacotes e patches precisam ser baixados para construir um sistema LFS e como armazená-los no novo sistema de arquivos. O Capítulo 4 discute a configuração de um ambiente de trabalho apropriado. Por favor, leia o Capítulo 4 cuidadosamente, uma vez que ele explica vários assuntos importantes a respeito dos quais você deveria estar ciente antes de começar seu trabalho ao longo do Capítulo 5 e além.

O Capítulo 5, explica a instalação do conjunto inicial de ferramentas, (binutils, gcc e glibc) usando técnicas de compilação cruzada para isolar as novas ferramentas das do sistema anfitrião.

O Capítulo 6 te mostra como compilar cruzadamente utilitários básicos usando o recém construído conjunto cruzado de ferramentas.

O Capítulo 7 então entra em um ambiente "chroot" onde nós usamos as novas ferramentas para construir todos o restante das ferramentas necessárias para criar o sistema LFS.

Esse esforço para isolar o sistema novo do sistema anfitrião possivelmente pareça excessivo. Uma explicação técnica completa do porquê isso é feito é fornecida nas Observações Técnicas do Conjunto de Ferramentas.

No Capítulo 8, o sistema LFS completo é construído. Outra vantagem fornecida pelo ambiente chroot é a de que ele te permite continuar usando o sistema anfitrião enquanto o LFS está sendo construído. Enquanto espera por compilações de pacotes completarem, você pode continuar usando seu computador normalmente.

Para finalizar a instalação, a configuração básica do sistema é concluída no Capítulo 9, e o núcleo e carregador de inicialização são criados no Capítulo 10. O Capítulo 11 contém informação sobre como continuar a experiência LFS além deste livro. Após os passos neste capítulo terem sido implementados, o computador estará pronto para reinicializar no novo sistema LFS.

Esse é o processo em poucas palavras. Informação detalhada sobre cada passo é apresentada nos capítulos seguintes. Itens que pareçam complicados agora serão esclarecidos e tudo ficará em seu devido lugar conforme você embarcar na aventura do LFS.

1.2. O que há de novo desde o lançamento mais recente

Aqui está uma lista dos pacotes atualizados desde o lançamento anterior do LFS.

Atualizado para:

-
- Acl-2.4.0
- Attr-2.6.0
- Autoconf-2.73
- Binutils-2.47
- Coreutils-9.11
- E2fsprogs-1.47.4
- Expat-2.8.3
- File-5.48
- Findutils-4.11.0
- Flit-Core-4.0.2
- Gawk-5.4.1
- GCC-16.2.0
- Glibc-2.44
- Groff-1.24.1
- Iana-Etc-20260805
- Inetutils-2.8
- IPRoute2-7.1.0
- Kbd-2.10.0
- Less-704
- Libcap-2.78
- Libelf originário do Elfutils-0.195
- Libffi-3.8.0
- Libtool-2.6.2
- Linux-7.1.8
- Man-pages-6.18
- Meson-1.12.0
- MPC-1.4.1
- OpenSSL-4.0.1
- Packaging-26.3
- Perl-5.44.0
- Pkgconf-3.0.5
- Procps-ng-4.0.7
- Python-3.14.7
- Sed-4.10
- Setuptools-84.0.0
- Shadow-4.20.2
- Sqlite-3530400
- Systemd-261.2
- Tcl-8.6.18

- Texinfo-7.3
- Tzdata-2026c
- Util-linux-2.42.2
- Vim-9.2.1025
- Wheel-0.48.0
- Xz-5.8.3

Adicionado:

- mpdecimal-4.0.1
- tar-1.35-acl_fix-1.patch
- Python-3.14.6-consolidated_fixes-1.patch

Removido:

-
- XML-Parser (Movido para BLFS)
- intltool (Movido para BLFS)

1.3. Registro das Mudanças

Esta é a versão 13.1-systemd do livro Linux From Scratch, datada de 01 de setembro de 2026. Se este livro for mais antigo que seis meses, uma versão mais nova e melhor provavelmente já está disponível. Para descobrir, por favor verifique um dos espelhos via <https://www.linuxfromscratch.org/mirrors.html>.

Abaixo está uma lista das mudanças feitas desde o lançamento anterior do livro.

Entradas do Registro das Mudanças:

- 2026-09-01
 - [bdubbs] - LFS-13.1 lançado.
- 2026-08-31
 - [bdubbs] - Atualização para vim-9.2.1025 (atualização de segurança). Corrige #6006.
- 2026-08-15
 - [bdubbs] - Atualização para expat-2.8.3 (atualização de segurança). Corrige #5996.
 - [bdubbs] - Atualização para iana-etc-20260805. Endereça #5006.
 - [bdubbs] - Atualização para libffi-3.8.0. Corrige #5993.
 - [bdubbs] - Atualização para linux-7.1.8. Corrige #5997.
 - [bdubbs] - Atualização para meson-1.12.0. Corrige #5998.
 - [bdubbs] - Atualização para procps-ng-4.0.7. Corrige #6000.
 - [bdubbs] - Atualização para setuptools-84.0.0 (módulo Python). Corrige #5994.
 - [bdubbs] - Atualização para shadow-4.20.2. Corrige #5995.
 - [bdubbs] - Atualização para vim-9.2.0954. Endereça #4500.
 - [bdubbs] - Atualização para wheel-0.48.0 (módulo Python). Corrige #5999.
- 2026-08-07
 - [bdubbs] - Atualização para flit_core-4.0.2 (módulo Python). Corrige #5989.
 - [bdubbs] - Atualização para gcc-16.2.0. Corrige #5992.

- [bdubbs] - Atualização para linux-7.1.7. Corrige #5988.
- [bdubbs] - Atualização para packaging-26.3 (módulo Python). Corrige #5990.
- [bdubbs] - Atualização para pkgconf-3.0.5. Corrige #5987.
- [bdubbs] - Atualização para Python-3.14.7 (atualização de segurança). Corrige #5991.
- 2026-08-01
 - [bdubbs] - Atualização para shadow-4.20.0. Corrige #5986.
 - [bdubbs] - Atualização para vim-9.2.0858 (atualização de segurança). Corrige #5982.
 - [bdubbs] - Atualização para systemd-261.2. Corrige #5981.
 - [bdubbs] - Atualização para sqlite-3.53.4. Corrige #5984.
 - [bdubbs] - Atualização para pkgconf-3.0.4. Corrige #5977.
 - [bdubbs] - Atualização para perl-5.44.0. Corrige #5980.
 - [bdubbs] - Atualização para linux-7.1.5. Corrige #5979.
 - [bdubbs] - Atualização para libtool-2.6.2. Corrige #5978.
 - [bdubbs] - Atualização para iana-etc-20260723. Endereça #5006.
 - [bdubbs] - Atualização para glibc-2.44 (atualização de segurança). Corrige #5983.
 - [bdubbs] - Atualização para binutils-2.47. Corrige #5985.
- 2026-07-15
 - [bdubbs] - Adicionar mpdecimal-4.0.1. Corrige #5963.
 - [bdubbs] - Atualização para tzdata-2026c. Corrige #5971.
 - [bdubbs] - Atualização para setuptools-83.0.0. Corrige #5969.
 - [bdubbs] - Atualização para pkgconf-3.0.2. Corrige #5962.
 - [bdubbs] - Atualização para meson-1.11.2. Corrige #5975.
 - [bdubbs] - Atualização para libffi-3.7.1. Corrige #5970.
 - [bdubbs] - Atualização para linux-7.1.3. Corrige #5968.
 - [bdubbs] - Atualização para gawk-5.4.1. Corrige #5973.
 - [bdubbs] - Atualização para findutils-4.11.0. Corrige #5974.
- 2026-07-01
 - [bdubbs] - Adicionar tar-1.35-acl_fix-1.patch. Endereça problema emergente originário do acl-2.4.0.
 - [bdubbs] - Adicionar correções consolidadas do Python3-3.14.6 (remendo de segurança). Corrige #5961.
 - [bdubbs] - Atualização para attr-2.6.0 (atualização de segurança). Corrige #5967.
 - [bdubbs] - Atualização para acl-2.4.0 (atualização de segurança). Corrige #5966.
 - [bdubbs] - Atualização para expat-2.8.2 (atualização de segurança). Corrige #5964.
 - [bdubbs] - Atualização para iana-etc-20260617. Endereça #5006.
 - [bdubbs] - Atualização para iproute2-7.1.0. Corrige #5954.
 - [bdubbs] - Atualização para libffi-3.6.0. Corrige #5958.
 - [bdubbs] - Atualização para linux-7.1.1. Corrige #5953.
 - [bdubbs] - Atualização para sqlite-autoconf-3.53.3. Corrige #5965.
 - [bdubbs] - Atualização para systemd-261. Corrige #5959.

- [bdubbs] - Atualização para util-linux-2.42.2 (atualização de segurança). Corrige #5955.
- [bdubbs] - Atualização para vim-9.2.0752 (atualização de segurança). Corrige #5956.
- 2026-06-15
 - [bdubbs] - Atualização para binutils-2.46.1. Corrige #5950.
 - [bdubbs] - Atualização para file-5.48. Corrige #5949.
 - [bdubbs] - Atualização para less-704. Corrige #5948.
 - [bdubbs] - Atualização para linux-7.0.12. Corrige #5945.
 - [bdubbs] - Atualização para openssl-4.0.1. Corrige #5951.
 - [bdubbs] - Atualização para Python3-3.14.6. Corrige #5952.
 - [bdubbs] - Atualização para sqlite-autoconf-3.53.2. Corrige #5946.
 - [bdubbs] - Atualização para vim-9.2.0640 (atualização de segurança). Corrige #5947.
- 2026-06-01
 - [bdubbs] - Atualização para systemd-260.2. Corrige #5944.
 - [bdubbs] - Atualização para kbd-2.10.0. Corrige #5943.
 - [bdubbs] - Atualização para util-linux-2.42.1. Corrige #5939.
 - [bdubbs] - Atualização para less-702. Corrige #5941.
 - [bdubbs] - Atualização para vim-9.2.0567 (atualização de segurança). Corrige #5940.
 - [bdubbs] - Atualização para iana-etc-20260529. Endereça #5006.
- 2026-05-24
 - [bdubbs] - Atualização para linux-7.0.10 (atualização de segurança). Corrige #5942.
- 2026-05-20
 - [bdubbs] - Atualização para linux-7.0.9 (atualização de segurança). Corrige #5928.
- 2026-05-18
 - [xry111] - Corrigir CVE-2026-4046, CVE-2026-5450 e CVE-2026-5928 na glibc (atualização de segurança). Corrige #5930.
- 2026-05-14
 - [renodr] - Atualização para expat-2.8.1 (atualização de segurança). Corrige #5933.
 - [renodr] - Atualização para tcl-8.6.18. Corrige #5935.
 - [renodr] - Atualização para vim-9.2.0480 (atualização de segurança). Corrige #5928.
 - [renodr] - Atualização para linux-7.0.7 (atualização de segurança). Corrige #5932.
 - [renodr] - Corrigir CVE-2026-7210 e CVE-2026-8328 no Python (atualização de segurança). Corrige #5934.
- 2026-05-11
 - [bdubbs] - Atualização para vim-9.2.0461 (atualização de segurança). Corrige #5938.
 - [bdubbs] - Atualização para sqlite-3.53.1. Corrige #5929.
 - [bdubbs] - Atualização para Python-3.14.5. Corrige #5931.
 - [bdubbs] - Atualização para openssl-4.0.0. Corrige #5912.
 - [bdubbs] - Atualização para linux-7.0.5. Corrige #5917.
 - [bdubbs] - Atualização para iana-etc-20260504. Endereça #5006.
 - [bdubbs] - Adicionar remendo de construção linux7 do Python-3.14.5. Endereça #5931.

- [bdubbs] - Adicionar remendo de correções do linux7 da glibc.
- [bdubbs] - Adicionar remendo de construção openssl4 do systemd. Endereça #5912.
- 2026-05-08
 - [bdubbs] - Atualização para gcc-16.1.0.. Corrige #5927.
- 2026-05-01
 - [bdubbs] - Atualização para wheel-0.47.0 (módulo Python). Corrige #5922.
 - [bdubbs] - Atualização para tzdata2026b. Corrige #5923.
 - [bdubbs] - Atualização para sed-4.10. Corrige #5921.
 - [bdubbs] - Atualização para packaging-26.2 (módulo Python). Corrige #5913.
 - [bdubbs] - Atualização para mpc-1.4.1. Corrige #5916.
 - [bdubbs] - Atualização para meson-1.11.1. Corrige #5919.
 - [bdubbs] - Atualização para vim-9.2.0421 (atualização de segurança). Corrige #5914.
 - [bdubbs] - Atualização para man-pages-6.18. Corrige #5920.
 - [bdubbs] - Atualização para iproute2-7.0.0. Corrige #5925.
 - [bdubbs] - Atualização para inetutils-2.8. Corrige #5926.
 - [bdubbs] - Atualização para expat-2.8.0 (atualização de segurança). Corrige #5924.
 - [bdubbs] - Atualização para elfutils-0.195 (libelf). Corrige #5915.
 - [bdubbs] - Atualização para coreutils-9.11. Corrige #5918.
 - [bdubbs] - Atualização para iana-etc-20260424. Endereça #5006.
- 2026-04-15
 - [jlocash] - Atualizar o remendo do Python-3.14.4 para 2 problemas mais (atualização de segurança). Endereça #5910.
- 2026-04-15
 - [bdubbs] - Atualização para meson-1.11.0. Corrige #5911.
 - [bdubbs] - Mover intltool e XML-Parser para o BLFS. Corrige #4964.
 - [bdubbs] - Atualização para vim-9.2.0340 (atualização de segurança). Corrige #5904.
 - [bdubbs] - Atualização para util-linux-2.42 (atualização de segurança). Corrige #5900.
 - [bdubbs] - Atualização para sqlite-3.53.0. Corrige #5909.
 - [bdubbs] - Atualização para Python3-3.14.4 (atualização de segurança). Corrige #5907.
 - [bdubbs] - Atualização para openssl-3.6.2 (atualização de segurança). Corrige #5906.
 - [bdubbs] - Atualização para linux-6.19.12. Corrige #5903.
 - [bdubbs] - Atualização para libcap-2.78 (atualização de segurança). Corrige #5905.
 - [bdubbs] - Atualização para iana-etc-20260409. Endereça #5006.
- 2026-04-04
 - [zeckma] - Fornece suporte de inicialização UEFI. Corrige #5819.
- 2026-04-01
 - [bdubbs] - Atualização para xz-5.8.3 (atualização de segurança). Corrige #5899.
 - [bdubbs] - Atualização para XML-Parser-2.54 (atualização de segurança). Corrige (novamente) #5892.

- [bdubbs] - Atualização para vim-9.2.0272. Endereça #4500.
- [bdubbs] - Atualização para systemd-260.1. Corrige #5890.
- [bdubbs] - Atualização para perl-5.42.2. Corrige #5898.
- [bdubbs] - Atualização para mpc-1.4.0. Corrige #5894.
- [bdubbs] - Atualização para meson-1.10.2. Corrige #5889.
- [bdubbs] - Atualização para linux-6.19.10. Corrige #5893.
- [bdubbs] - Atualização para iana-etc-20260327. Endereça #5006.
- [bdubbs] - Atualização para groff-1.24.1. Corrige #5888.
- [bdubbs] - Atualização para autoconf-2.73. Corrige #5897.
- 2026-03-26
 - [bdubbs] - Atualização para XML-Parser-2.53 (atualização de segurança). Corrige #5892.
 - [bdubbs] - Adicionar um sed ao glibc-2.43 (atualização de segurança). Corrige #5895.
 - [bdubbs] - Adicionar um remendo ao Python-3.14.3 (atualização de segurança). Corrige #5896.
 - [bdubbs] - Atualização para expat-2.7.5 (atualização de segurança). Corrige #5891.
- 2026-03-15
 - [bdubbs] - Atualização para vim-9.2.0161. Endereça #4500.
 - [bdubbs] - Atualização para tzdata-2026a. Corrige #5882.
 - [bdubbs] - Atualização para texinfo-7.3. Corrige #5883.
 - [bdubbs] - Atualização para systemd-259.5. Corrige #5879.
 - [bdubbs] - Atualização para sqlite-autoconf-3510300 (3.51.3). Corrige #5885.
 - [bdubbs] - Atualização para shadow-4.19.4. Corrige #5881.
 - [bdubbs] - Atualização para setuptools-82.0.1 (módulo Python). Corrige #5887.
 - [bdubbs] - Atualização para perl-5.42.1. Corrige #5886.
 - [bdubbs] - Atualização para linux-6.19.8. Corrige #5867.
 - [bdubbs] - Atualização para iproute2-6.19.0. Corrige #5874.
 - [bdubbs] - Atualização para iana-etc-20260306. Endereça #5006.
 - [bdubbs] - Atualização para groff-1.24.0. Corrige #5880.
 - [bdubbs] - Atualização para gawk-5.4.0. Corrige #5876.
 - [bdubbs] - Atualização para file-5.47. Corrige #5878.
 - [bdubbs] - Atualização para e2fsprogs-1.47.4. Corrige #5884.
- 2026-03-05
 - [bdubbs] - LFS-13.0 lançado.

1.4. Recursos

1.4.1. Perguntas Frequentes

Se durante a construção do sistema LFS você encontrar quaisquer erros, tiver quaisquer perguntas, ou entender que há um erro de digitação no livro, [então], por favor, comece consultando as Perguntas Feitas Frequentemente (FAQ) que estão localizadas em <https://www.linuxfromscratch.org/faq/>.

1.4.2. Listas de Correio Eletrônico

O servidor [linuxfromscratch.org](https://www.linuxfromscratch.org) hospeda um número de listas de discussão usadas para o desenvolvimento do projeto LFS. Essas listas incluem as principais listas de desenvolvimento e suporte, dentre outras. Se você não conseguir encontrar uma resposta para o seu problema na página do FAQ, [então] o próximo passo seria procurar nas listas de discussão em <https://www.linuxfromscratch.org/search.html>.

Para informação sobre as diversas listas, como se inscrever, localização de arquivos e informações adicionais, visite <https://www.linuxfromscratch.org/mail.html>.

1.4.3. IRC

Vários membros da comunidade LFS oferecem assistência via Internet Relay Chat (IRC). Antes de usar esse suporte, por favor certifique-se de que sua pergunta já não foi respondida no FAQ do LFS ou nos arquivos das listas de discussão. Você pode encontrar a rede de comunicação IRC em [irc.libera.chat](https://libera.chat). O canal de suporte é chamado de `#lfs-support`.

1.4.4. Sítios Espelho

O projeto LFS tem um número de espelhos mundo afora para tornar o acesso ao sítio do projeto e o download dos pacotes exigidos mais conveniente. Por favor visite o sítio web do LFS em <https://www.linuxfromscratch.org/mirrors.html> para uma lista dos espelhos atuais.

1.4.5. Informação de Contato

Por favor, direcione todas as suas perguntas e comentários para uma das listas de discussão do LFS (veja acima).

1.5. Ajuda



Nota

Caso você tenha encontrado um problema ao construir um pacote com a instrução do LFS, nós desencorajamos fortemente postar o problema diretamente no canal de suporte do fluxo de desenvolvimento antes de discutir por meio de um canal de suporte do LFS listado em Seção 1.4, “Recursos.” Fazer isso frequentemente é bastante ineficiente porque os(as) mantenedores(as) de fluxo de desenvolvimento raramente estão familiarizados com o procedimento de construção do LFS. Mesmo se você realmente tiver encontrado um problema de fluxo de desenvolvimento, a comunidade LFS ainda consegue ajudar a isolar as informações desejadas pelos(as) mantenedores(as) de fluxo de desenvolvimento e produzir um informe adequado.

Se precisar fazer uma pergunta diretamente por meio de um canal de suporte do(a) desenvolvedor(a), [então] você deveria observar, pelo menos, que muitos projetos de desenvolvedor(a) tem os canais de suporte separados do rastreador de defeitos. Os informes de “defeito” para fazer perguntas são considerados inválidos e podem incomodar os(as) desenvolvedores(as) upstream para esses projetos.

Se um problema ou uma pergunta for encontrado durante o trabalho ao longo deste livro, [então], por favor, verifique a página de Perguntas Frequentes em <https://www.linuxfromscratch.org/faq/#generalfaq>. Perguntas frequentemente já estão respondidas lá. Se sua pergunta não estiver respondida nessa página, [então], por favor, tente encontrar a origem do problema. A dica seguinte te dará alguma orientação com relação à resolução de problemas: <https://www.linuxfromscratch.org/hints/downloads/files/errors.txt>.

Se você não conseguir achar seu problema listado nas Perguntas Frequentes, [então] procure nas listas de discussão em <https://www.linuxfromscratch.org/search.html>.

Nós também temos uma comunidade LFS maravilhosa que está disposta a oferecer assistência por meio das listas de discussão e IRC (veja a seção Seção 1.4, “Recursos” deste livro). Entretanto, nós temos várias perguntas de suporte todos os dias e muitas delas poderiam ter sido facilmente respondidas indo para as Perguntas Frequentes ou procurando nas listas de discussão primeiro. Então, para que nós possamos oferecer a melhor assistência possível, você deveria primeiro fazer alguma pesquisa por conta própria. Isso nos permite focar nas necessidades menos usuais de suporte. Se suas buscas não produzirem uma solução, [então], por favor, inclua todas as informações relevantes (mencionadas abaixo) no seu pedido por ajuda.

1.5.1. Coisas a Mencionar

Além de uma breve explicação do problema sendo vivenciado, qualquer solicitação por ajuda deveria incluir estas coisas essenciais:

- A versão do livro sendo usada (neste caso 13.1-systemd)
- A distribuição anfitriã e versão sendo usada para criar o LFS
- A saída gerada originária do script Exigências do Sistema Anfitrião
- O pacote ou seção onde o problema foi encontrado
- A mensagem exata do erro ou uma descrição clara do problema
- Observação se você tiver se desviado do livro afinal



Nota

Desviar-se deste livro *não* significa que nós não vamos te ajudar. Afinal de contas, o LFS é acerca de preferência pessoal. Ser sincero a respeito de quaisquer mudanças no procedimento estabelecido nos ajuda a avaliar e determinar possíveis causas do seu problema.

1.5.2. Problemas do Script de Configuração

Se algo der errado quando executar o script **configure**, [então] revise o arquivo `config.log`. Esse arquivo possivelmente contenha erros encontrados durante o **configure** os quais não foram exibidos na tela. Inclua as linhas *relevantes* se você precisar pedir ajuda.

1.5.3. Problemas de Compilação

Tanto a saída gerada da tela quanto o conteúdo de vários arquivos são úteis para determinar a causa de problemas de compilação. A saída gerada da tela originária do script **configure** e a execução do **make** podem ser úteis. Não é necessário incluir a saída gerada inteira, mas inclua toda a informação relevante. Aqui está um exemplo do tipo de informação a incluir a partir da saída gerada de tela do **make**.

```
gcc -D ALIASEPATH="/mnt/lfs/usr/share/locale:."
-D LOCALEDIR="/mnt/lfs/usr/share/locale"
-D LIBDIR="/mnt/lfs/usr/lib"
-D INCLUDEDIR="/mnt/lfs/usr/include" -D HAVE_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
```

Nesse caso, muitas pessoas incluiriam apenas a seção final:

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

Essa não é informação suficiente para diagnosticar o problema, pois essa linha apenas mostra que algo deu errado, não *o que* deu errado. A seção inteira, como no exemplo acima, é o que deveria ser salva, porque ela inclui o comando que foi executado e todas as mensagens de erro associadas.

Um artigo excelente sobre como pedir ajuda na Internet está disponível online em <http://catb.org/~esr/faqs/smart-questions.html>. Leia esse documento e siga as dicas. Fazer isso aumentará a possibilidade de obter a ajuda que você precisa.

Parte II. Preparando para a Construção

Capítulo 2. Preparando o Sistema Anfitrião

2.1. Introdução

Neste capítulo, as ferramentas do anfitrião necessárias para a construção do LFS são verificadas e, se necessário, instaladas. Então uma partição que hospedará o sistema LFS é preparada. Nós criaremos a própria partição, criaremos um sistema de arquivos nela e a montaremos.

2.2. Exigências do Sistema Anfitrião

2.2.1. Hardware

Os(As) editores(as) do LFS recomendam que a CPU do sistema tenha pelo menos quatro núcleos e que o sistema tenha pelo menos oito (08) GB de memória. Os sistemas mais antigos que não atendam a essas exigências ainda funcionarão, porém o tempo para construir os pacotes será significativamente maior que o documentado.

2.2.2. Software

O teu sistema anfitrião deveria ter o seguinte logiciário com as versões mínimas indicadas. Isso não deveria ser um problema para a maioria das distribuições modernas do Linux. Também, observe que muitas distribuições colocarão cabeçalhos de logiciários dentro de pacotes separados, frequentemente na forma de <nome-pacote>-devel ou <nome-pacote>-dev. Certifique-se de instalá-los se a tua distribuição os fornecer.

Versões anteriores dos pacotes de software listados possivelmente funcionem, porém não foram testados.



Atenção

Esteja ciente que GCC mais recente que 16.2.0 ou Binutils mais recente que 2.47 não foram testadas. Um GCC mais recente é altamente provável quebrar a construção deste lançamento do LFS. Não informe tal quebra para os(as) editores(as) do LFS. Se tua distribuição anfitriã tiver versões mais recentes desses pacotes, você tem várias opções:

- Usar um lançamento mais recente do livro LFS, se existir um
- Usar uma distribuição anfitriã mais antiga
- Desatualizar o GCC se isso for suportado pela distribuição anfitriã
- Usar o livro mais recente de desenvolvimento do LFS. As atualizações para a versão mais recente dos pacotes no LFS geralmente são adicionadas dentro de duas a quatro semanas depois do lançamento.

- **Bash-3.2** (/bin/sh deveria ser um link simbólico ou real para bash)
- **Binutils-2.13.1** (Versões maiores que 2.47 possivelmente não funcionem dado que elas não foram testadas)
- **Bison-2.7** (/usr/bin/yacc deveria ser um link para bison ou script pequeno que executa bison)
- **GNU Coreutils-8.1** ou **Utils Coreutils-0.8**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk deveria ser um link para gawk)
- **GCC-5.4** incluindo o compilador C++, g++ (Versões maiores que 16.2.0 possivelmente não funcionem dado que elas não foram testadas). As bibliotecas C e C++ padrão (com cabeçalhos) também precisam estar presentes, de modo que o compilador C++ consiga construir programas hospedados
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Núcleo Linux-5.10**

A razão para a exigência da versão do núcleo é a de que nós especificamos essa versão quando da construção da glibc no Capítulo 5 e Capítulo 8, de forma que as soluções alternativas para os núcleos mais antigos não estão habilitadas e a glibc compilada é ligeiramente mais rápida e menor. Em dezembro de 2024, 5.10 é o lançamento mais antigo do núcleo ainda suportado pelos(as) desenvolvedores(as) do núcleo. Alguns lançamentos de núcleo mais antigos que 5.10 possivelmente ainda sejam suportados por equipes de terceiros, porém não são considerados lançamentos oficiais de desenvolvedor(a) do núcleo; leia-se <https://kernel.org/category/releases.html> para os detalhes.

Se o núcleo do anfitrião for anterior a 5.10, [então] você precisará substituir o núcleo por uma versão mais atualizada. Existem duas maneiras de você fazer isso. Primeira, veja se seu fornecedor Linux fornece um pacote do núcleo 5.10 ou mais atual. Se sim, [então] você possivelmente deseje instalá-lo. Se seu fornecedor não oferecer um pacote de núcleo aceitável ou você preferisse não instalá-lo, [então] você mesmo(a) pode compilar um núcleo. Instruções para a compilação de núcleo e configuração de carregador de inicialização (presumindo que o anfitrião usa GRUB) estão localizadas no Capítulo 10.

Nós exigimos que o núcleo do anfitrião suporte o pseudo terminal UNIX 98 (PTY). Ele deveria estar habilitado em todas as distribuições desktop ou servidor que embarquem o Linux 5.10 ou um núcleo mais recente. Se você estiver construindo um núcleo personalizado de anfitrião, certifique-se de que CONFIG_UNIX98_PTYS esteja configurada como y na configuração do núcleo.

- **M4-1.4.10**
- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**
- **Texinfo-5.0**
- **Xz-5.0.0**

Além disso, se você precisar criar uma nova ESP (EFI System Partition, leia Seção 2.5, “Criando um Sistema de Arquivos na Partição” para detalhes), *dosfstools* é necessário.



Importante

Perceba que os links simbólicos mencionados acima são exigidos para construir um sistema LFS usando as instruções contidas neste livro. Links simbólicos que apontem para outro software (tais como dash, mawk, etc.) possivelmente funcionem, porém não são testados ou suportados pela equipe de desenvolvimento do LFS e possivelmente exijam ou desvio das instruções ou remendos adicionais para alguns pacotes.

Para ver se o seu sistema anfitrião tem todas as versões apropriadas e a habilidade de compilar aplicativos, execute os seguintes comandos:

```
cat > version-check.sh << "EOF"
#!/bin/bash
# A script to list version numbers of critical development tools

# If you have tools installed in other directories, adjust PATH here AND
# in ~/.lfs/.bashrc (section 4.4) as well.

LC_ALL=C
PATH=/usr/bin:/bin

bail() { echo "FATAL: $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep does not work"
sed '' /dev/null || bail "sed does not work"
sort /dev/null || bail "sort does not work"
```

```

ver_check()
{
    if ! type -p $2 >/dev/null
    then
        echo "ERROR: Cannot find $2 ($1)"; return 1;
    fi
    v=$(($2 --version 2>&1 | grep -E -o '[0-9]+\.[0-9\.]?[a-z]*' | head -n1)
    if printf '%s\n' $3 $v | sort --version-sort --check >/dev/null
    then
        printf "OK:    %-9s %-6s >= $3\n" "$1" "$v"; return 0;
    else
        printf "ERROR: %-9s is TOO OLD ($3 or later required)\n" "$1";
        return 1;
    fi
}

ver_kernel()
{
    kver=$(uname -r | grep -E -o '^[0-9\.]+' )
    if printf '%s\n' $1 $kver | sort --version-sort --check >/dev/null
    then
        printf "OK:    Linux Kernel $kver >= $1\n"; return 0;
    else
        printf "ERROR: Linux Kernel ($kver) is TOO OLD ($1 or later required)\n" "$kver";
        return 1;
    fi
}

# Coreutils first because --version-sort needs Coreutils >= 7.0
if sort --version |& grep -q utils; then
    ver_check Coreutils sort      0.8 || bail "Utils Coreutils too old, stop"
else
    ver_check Coreutils sort      8.1 || bail "GNU Coreutils too old, stop"
fi
ver_check Bash                bash      3.2
ver_check Binutils            ld        2.13.1
ver_check Bison               bison     2.7
ver_check Diffutils           diff     2.8.1
ver_check Findutils           find     4.2.31
ver_check Gawk                gawk     4.0.1
ver_check GCC                 gcc       5.4
ver_check "GCC (C++)"         g++     5.4
ver_check Grep                grep     2.5.1a
ver_check Gzip                gzip     1.3.12
ver_check M4                  m4      1.4.10
ver_check Make                make     4.0
ver_check Patch               patch    2.5.4
ver_check Perl                perl     5.8.8
ver_check Python              python3  3.4
ver_check Sed                 sed      4.1.5
ver_check Tar                 tar      1.22
ver_check Texinfo             texi2any 5.0
ver_check Xz                  xz      5.0.0
ver_kernel 5.10

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK:    Linux Kernel supports UNIX 98 PTY";
else echo "ERROR: Linux Kernel does NOT support UNIX 98 PTY"; fi

alias_check() {
    if $1 --version 2>&1 | grep -qi $2
    then printf "OK:    %-4s is $2\n" "$1";
    else printf "ERROR: %-4s is NOT $2\n" "$1"; fi
}
echo "Aliases:"
alias_check awk GNU

```

```
alias_check yacc Bison
alias_check sh Bash

echo "Compiler check:"
if printf "int main(){}" | g++ -x c++ -
then echo "OK:      g++ works";
else echo "ERROR: g++ does NOT work"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
    echo "ERROR: nproc is not available or it produces empty output"
else
    echo "OK: nproc reports $(nproc) logical cores are available"
fi
EOF

bash version-check.sh
```

2.3. Construindo o LFS em Estágios

O LFS é projetado para ser construído em uma sessão. Isto é, as instruções assumem que o sistema não será desligado durante o processo. Isso não significa que o sistema tenha de ser construído de uma vez só. O problema é que certos procedimentos precisam ser repetidos depois de uma inicialização quando se retomando o LFS em pontos diferentes.

2.3.1. Capítulos 1–4

Esses capítulos executam comandos no sistema anfitrião. Quando da reinicialização, esteja certo(a) de uma coisa:

- Os procedimentos realizados como o(a) usante *root* depois da Seção 2.4 precisam ter a variável de ambiente LFS configurada *PARA O(A) USANTE ROOT*.

2.3.2. Capítulos 5–6

- A partição */mnt/lfs* precisa estar montada.
- Esses dois capítulos *precisam* ser feitos como usante *lfs*. Um comando **su - lfs** precisa ser emitido antes de realizar alguma tarefa nesses capítulos. Se não fizer isso, você estará no risco de instalar pacotes no sistema anfitrião e potencialmente torná-lo desusável.
- Os procedimentos em Instruções Gerais de Compilação são críticos. Se existir qualquer dúvida se um pacote tiver sido instalado corretamente, [então] certifique-se de que o tarball anteriormente expandido tenha sido removido, então extraia novamente o pacote e complete todas as instruções naquela seção.

2.3.3. Capítulos 7–10

- A partição */mnt/lfs* precisa estar montada.
- Umass poucas operações, desde “Preparando Sistemas de Arquivos Virtuais do Núcleo” até “Entrando no Ambiente Chroot”, precisam ser feitas como o(a) usante *root*, com a variável de ambiente LFS configurada para o(a) usante *root*.
- Quando entrar em *chroot*, a variável de ambiente LFS precisa estar configurada para o(a) *root*. A variável LFS não é usada depois que o ambiente *chroot* tiver sido acessado.
- Os sistemas virtuais de arquivo precisam estar montados. Isso pode ser feito antes ou depois de entrar no *chroot* mudando-se para um terminal virtual do anfitrião e, como *root*, executando-se os comandos em Seção 7.3.1, “Montando e Povoando */dev*” e Seção 7.3.2, “Montando Sistemas de Arquivos Virtuais do Núcleo.”

2.4. Criando uma Nova Partição

Como a maior parte dos outros sistemas operacionais, o LFS geralmente é instalado em uma partição dedicada. A abordagem recomendada para construir um sistema LFS é a de usar uma partição disponível vazia ou, se você tiver espaço suficiente não particionado, criar uma.

A minimal system requires a partition of around 10 gigabytes (GB). This is enough to store all the source tarballs and compile the packages. However, if the LFS system is intended to be the primary Linux system, additional software will probably be installed which will require additional space. A 30 GB partition is a reasonable size to provide for growth. The LFS system itself will not take up this much room. A large portion of this requirement is to provide sufficient free temporary storage as well as for adding additional capabilities after LFS is complete. Additionally, compiling packages can require a lot of disk space which will be reclaimed after the package is installed.

Como nem sempre existe Memória de Acesso Aleatório (RAM) suficiente disponível para processos de compilação, é uma boa ideia usar uma pequena partição de disco como espaço de swap. Ele é usado pelo kernel para armazenar dados raramente usados e deixa mais memória disponível para processos ativos. A partição de swap para um sistema LFS pode ser a mesma que aquela usada pelo sistema anfitrião, caso no qual não é necessário criar outra.

Inicie um aplicativo de particionamento de disco como o "**cdisk**" ou o "**fdisk**" com uma opção de linha de comando indicando o disco rígido no qual a nova partição será criada—por exemplo `/dev/sda` para a unidade primária de disco. Crie uma partição nativa do "Linux" e uma partição "swap", se necessária. Por favor, recorra a "**cdisk(8)**" ou a "**fdisk(8)**" se você ainda não sabe como usar os aplicativos.



Nota

Para usantes experientes, outros esquemas de particionamento são possíveis. O novo sistema LFS pode estar em um vetor de software *RAID* ou em um volume lógico *LVM*. Entretanto, algumas dessas opções exigem um *initramfs*, o que é um tópico avançado. Essas metodologias de particionamento não são recomendadas para usantes iniciantes do LFS.

Lembre-se da designação da nova partição (por exemplo, `sda5`). Este livro se referirá a essa como a partição do LFS. Lembre-se também da designação da partição swap. Esses nomes serão necessários posteriormente para o arquivo `/etc/fstab`.

2.4.1. Outros Problemas de Partição

Solicitações de conselhos a respeito de particionamento do sistema frequentemente são postados nas listas de discussão do LFS. Esse é um tópico altamente subjetivo. O padrão para a maioria das distribuições é o de usar a unidade inteira com a exceção de uma pequena partição de swap. Isso não é ideal para o LFS por várias razões. Isso reduz flexibilidade; torna o compartilhamento de dados entre múltiplas distribuições ou construções do LFS mais difícil; torna as cópias de segurança mais demoradas; e podem desperdiçar espaço de disco devido à alocação ineficiente de estruturas do sistema de arquivos.

2.4.1.1. A Partição Raiz

Uma partição raiz do LFS (não confundir com o diretório `/root`) de vinte (20) gigabytes é uma boa escolha para a maior parte dos sistemas. Ela fornece espaço suficiente para construir o LFS e a maior parte do BLFS, mas é pequena o suficiente de forma que múltiplas partições podem ser criadas facilmente para experimentação.

2.4.1.2. A Partição Swap

A maioria das distribuições automaticamente cria uma partição swap. Geralmente o tamanho recomendado da partição swap é o de cerca de o dobro da quantidade de RAM física, entretanto isso raramente é necessário. Se o espaço de disco for limitado, [então] mantenha a partição swap com dois (2) gigabytes e monitore a quantidade de troca de disco.

Se você quiser usar o recurso da hibernação do Linux (suspend-to-disk), copia o conteúdo da RAM para a partição swap antes de desligar a máquina. Nesse caso o tamanho da partição swap deveria ser pelo menos tão grande quanto a RAM instalada do sistema.

O uso de swap nunca é bom. Para unidades rígidas mecânicas você geralmente pode dizer se um sistema está usando swap simplesmente monitorando a atividade do disco e observando como o sistema reage a comandos. Com um SSD você não estará apta(o) a monitorar swap, porém você consegue dizer quanto espaço de swap está sendo usado executando os aplicativos **top** ou **free**. O uso de um SSD para uma partição swap deveria ser evitado se possível. A primeira reação em caso de uso de swap deveria ser verificar se existe um comando irracional como tentar editar um arquivo de cinco gigabytes. Se o uso de swap se tornar uma ocorrência recorrente, [então] a melhor solução é a de comprar mais RAM para seu sistema.

2.4.1.3. A Partição de BIOS do Grub

Se o *disco de inicialização* tiver sido particionado com a Tabela de Partição GUID (GPT), então uma partição pequena, tipicamente um (1) MB, precisará ser criada se o sistema estiver sendo inicializado com BIOS e ela já não existir. Essa partição não é formatada, porém precisa estar disponível para o GRUB usar durante a instalação do carregador de inicialização. Essa partição normalmente será rotulada de 'BIOS Boot' se usar o **fdisk** ou terá um código de *EF02* se usar o comando **gdisk**.

Se o disco de inicialização estiver particionado com uma Tabela MBR de Partição ou um rótulo DOS de disco, então essa partição não será necessária, pois já existe espaço antes da primeira partição que o Grub consegue usar.



Nota

A Partição de BIOS do Grub precisa estar na unidade que o BIOS usa para inicializar o sistema. Essa não é necessariamente a unidade que mantém a partição raiz do LFS. Os discos em um sistema possivelmente usem tipos diferentes de tabela de partição. A necessidade da partição de BIOS do Grub depende somente do tipo de tabela de partição do disco de inicialização.

2.4.1.4. A Partição EFI de Sistema

Essa partição, também conhecida como a *ESP*, é necessária ao inicializar o sistema com UEFI. Ela armazena o aplicativo EFI que é executado durante a inicialização. A unidade de inicialização pode ser particionada com a Tabela MBR de Partição ou DOS, mas problemas de compatibilidade tenderão a surgir como um resultado. Portanto, é sempre uma boa ideia, nesse caso, particionar a unidade de inicialização com uma Tabela GUID de Partição (GPT). Se você estiver inicializando somente o LFS a partir da partição, 20 MB ou menos podem ser suficientes. A partição deveria ser maior que o tamanho da imagem da EFI, porque o GRUB despeja muitos dados na partição antes de criar a imagem da EFI. Para maior segurança, de 128 MB a 256 MB é recomendado, mas pode ser reduzida consideravelmente com alguma experimentação. O rótulo da partição deveria ser 'EFI System' se usar **fdisk**.

Para Grub, a Partição EFI de Sistema deveria estar localizada em `/boot/efi`.

Muitos sistemas UEFI têm um Módulo de Suporte de Compatibilidade (MSC) ou opção de Inicialização Legada, permitindo inicializar com BIOS. Poderia ser uma boa ideia criar uma partição de BIOS do Grub se teu sistema suportar MSC, caso a inicialização UEFI não funcione como esperado.

2.4.1.5. Partições de Conveniência

Existem várias outras partições que não são exigidas, porém deveriam ser consideradas ao se projetar um layout de disco. A lista seguinte não é abrangente, mas é entendida como um guia.

- `/boot` – Altamente recomendada. Use essa partição para armazenar os kernels e outras informações de inicialização. Para minimizar potenciais problemas de inicialização com discos maiores, torne essa a primeira partição física na sua primeira unidade de disco. Um tamanho de partição de duzentos (200) megabytes é adequado.

- `/home` – Altamente recomendada. Compartilhe seu diretório lar e personalizações de usuário entre múltiplas distribuições ou construções do LFS. O tamanho geralmente é bastante grande e depende do espaço de disco disponível.
- `/usr` – No LFS, `/bin`, `/lib` e `/sbin` são links simbólicos para seus homólogos em `/usr`. Assim `/usr` contém todos os binários necessários para o sistema executar. Para o LFS, uma partição separada para `/usr` normalmente não é necessária. Se, de qualquer maneira, você criá-la, [então] você deveria tornar uma partição grande o suficiente para acomodar todos os aplicativos e bibliotecas no sistema. A partição raiz pode ser bem pequena (talvez apenas um gigabyte) nessa configuração, de forma que ela seja adequada para um "thin client" ou estação de trabalho sem disco (onde `/usr` é montado a partir de um servidor remoto). Entretanto, você deveria estar ciente de que um `initramfs` (não coberto pelo LFS) será necessário para inicializar um sistema com partição `/usr` separada.
- `/opt` – Esse diretório é mais útil para o BLFS onde múltiplos pacotes grandes como o KDE ou o Texlive podem ser instalados sem embutir os arquivos na hierarquia `/usr`. Se usado, 5 a 10 gigabytes geralmente é adequado.
- `/tmp` – Por padrão, o `systemd` monta um `tmpfs` aqui. Se você quiser substituir esse comportamento, siga Seção 9.10.3, "Desabilitando `tmpfs` para `/tmp`" quando configurar o sistema LFS.
- `/usr/src` – Essa partição é muito útil para disponibilizar um local para armazenar os arquivos fonte do BLFS e compartilhá-los entre construções do LFS. Ela também pode ser usada como um local para construir pacotes do BLFS. Uma partição razoavelmente grande de 30 a 50 gigabytes fornece abundância de espaço.

Qualquer partição separada que você queira montada automaticamente quando o sistema iniciar precisa ser especificada no arquivo `/etc/fstab`. Detalhes a respeito do como especificar partições serão discutidos na Seção 10.2, "Criando o Arquivo `/etc/fstab`".

2.4.2. Um Esquema de Disco de Exemplo

Abaixo, está um exemplo de esquema para uma unidade de disco vazia.

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	22527	10.0 MiB	EF00	EFI system partition
2	22528	24575	1024.0 KiB	EF02	BIOS boot partition
3	24576	1048575	500.0 MiB	8300	/boot
4	1048576	5242879	2.0 GiB	8200	swap
5	5242880	89128959	40.0 GiB	8300	lfs13.0+
6	89128960	173015039	40.0 GiB	8300	/home

O exemplo acima faz algumas presunções:

- A tabela de partições é uma GUID Partition Table (GPT).
- Ambas as partições de inicialização, EFI e BIOS, estão presentes, embora somente uma será usada. Qual será usada depende do firmware do sistema. Se o sistema for antigo, ele não terá recursos UEFI em absoluto. Alguns sistemas mais recentes conseguem desabilitar a UEFI por intermédio da configuração do firmware, desabilitando a "Inicialização Segura" e habilitando o "Suporte Legado" ou o "CSM" (Compatibility Support Module). Se você souber antecipadamente qual modo usará, a outra partição pode ser omitida.
- A partição EFI precisa ser formatada como VFAT.
- A partição do BIOS não está formatada.
- A partição de área de troca precisa ser inicializada.
- A partição `/boot` pode ser formatada como `ext2`, já que ela raramente é escrita (e mesmo assim, somente pelo(a) root) e não precisa de um diário.
- A recomendação para todas as outras partições é a de usar formatação `ext4`.
- Outra partição pode ser adicionada para instalar o sistema "anfitrião" para construir o LFS. Uma partição de tamanho mínimo, de 10 GiB, deveria ser suficiente. Se você estiver construindo o sistema usando um LiveCD, uma partição de anfitrião possivelmente não seja exigida.

2.5. Criando um Sistema de Arquivos na Partição

Uma partição é apenas um intervalo de setores em uma unidade de disco, delimitados por fronteiras configuradas em uma tabela de partição. Antes do sistema operacional conseguir usar uma partição para armazenar quaisquer arquivos, a partição precisa estar formatada para conter um sistema de arquivos, tipicamente consistindo de um rótulo, blocos de diretório, blocos de dados e um esquema de indexação para localizar um arquivo em particular mediante demanda. O sistema de arquivos também auxilia o SO a manter rastreio do espaço livre na partição; reservar os setores necessários quando um arquivo novo for criado ou um arquivo existente for estendido; e reciclar os segmentos de dados livres criados quando arquivos são deletados. Possivelmente também forneça suporte para redundância de dados e para recuperação dos erros.

O LFS consegue usar qualquer sistema de arquivos reconhecido pelo kernel Linux, mas os tipos mais comuns são ext3 e ext4. A escolha do sistema de arquivos certo pode ser complexa; depende das características dos arquivos e do tamanho da partição. Por exemplo:

- ext2
é adequado para partições pequenas que são atualizadas com pouca frequência tais como /boot.
- ext3
é uma atualização do ext2 que inclui um diário para ajudar a recuperar a situação da partição no caso de desligamento inadequado. É comumente usada como sistema de arquivos de propósito geral.
- ext4
é a versão mais recente da família ext de sistemas de arquivos. Ela fornece várias capacidades novas incluindo carimbos de tempo em nano segundos; criação e uso de arquivos muito grandes (até 16 TB); e melhoramentos de velocidade.

Outros sistemas de arquivos, incluindo FAT32, NTFS, JFS e XFS são úteis para propósitos especializados. Mais informação a respeito desses sistemas de arquivos, e de muitos outros, pode ser encontrada em https://en.wikipedia.org/wiki/Comparison_of_file_systems.

O LFS assume que o sistema de arquivos raiz (/) é do tipo ext4. Para criar um sistema de arquivos ext4 na partição do LFS, emita o seguinte comando:

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

Substitua <xxx> pelo nome da partição do LFS.

Se você estiver usando uma partição de swap existente, não existe necessidade de formatá-la. Se uma nova partição de swap foi criada, ela precisará ser inicializada com este comando:

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

Substitua <yyy> pelo nome da partição swap.

Se tiver criado uma Partição EFI de Sistema, você terá algumas opções. As placas-mãe, ao inicializar com UEFI, procuram aplicativos EFI em partições formatadas com uma variante de FAT (FAT12, FAT16, FAT32, VFAT, entre outras). Geralmente, FAT32 ou VFAT são as opções recomendadas.

Se você tiver escolhido FAT32, formate a ESP com o seguinte comando:

```
mkfs.fat -F 32 /dev/<yyy>
```

Por outro lado, se tiver escolhido VFAT, você pode executar o seguinte comando:

```
mkfs.vfat /dev/<zzz>
```

Substitua <zzz> pelo nome da Partição EFI de Sistema.

2.6. Configurando a Variável \$LFS e o Umask

Ao longo deste livro, a variável de ambiente `LFS` será usada muitas vezes. Você deveria se assegurar de que essa variável sempre está definida no decorrer do processo de construção do LFS. Ela deveria ser configurada para o nome do diretório onde você estará construindo seu sistema LFS - nós usaremos `/mnt/lfs` como um exemplo, porém você possivelmente escolha qualquer nome de diretório que queira. Se você está construindo o LFS em uma partição separada, esse diretório será o ponto de montagem para a partição. Escolha um local de diretório e configure a variável com o seguinte comando:

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

Ter essa variável configurada é benéfico naqueles comandos, tais como `mkdir -v $LFS/tools`, que podem ser digitados literalmente. O shell automaticamente substituirá “`$LFS`” por “`/mnt/lfs`” (ou para o que a variável foi configurada) quando processar a linha de comando.

Agora configure a máscara de criação do modo de arquivo (`umask`) para `022` caso a distribuição anfitriã use um padrão diferente:

```
umask 022
```

Configurar o `umask` como `022` garante que os arquivos e diretórios recém-criados sejam escrevíveis somente pelos(as) donos(as) deles, mas sejam legíveis e pesquisáveis (somente para diretórios) por qualquer pessoa (assumindo que os modos padrão são usados pela chamada de sistema `open(2)`, novos arquivos acabarão com o modo de permissão `644` e diretórios com o modo `755`). Um padrão excessivamente permissivo pode deixar buracos de segurança no sistema LFS, e um padrão excessivamente restritivo pode causar problemas estranhos ao construir ou usar o sistema LFS.



Cuidado

Não se esqueça de verificar se `LFS` está configurada e se o `umask` está configurado como `022` sempre que você sair e reentrar no ambiente de trabalho atual (como ao fazer um `su` para `root` ou outro(a) usante). Verifique se a variável `LFS` está configurada corretamente com:

```
echo $LFS
```

Certifique-se de que a saída gerada mostre o caminho para o local de construção do teu sistema LFS, que é `/mnt/lfs` se o exemplo fornecido foi seguido.

Verifique se o `umask` está configurado corretamente com:

```
umask
```

A saída gerada possivelmente seja `0022` ou `022` (o número de zeros à esquerda depende da distribuição anfitriã).

Se qualquer saída gerada desses dois comandos estiver incorreta, use o comando dado anteriormente nesta página para configurar `$LFS` como o nome correto de diretório e configurar `umask` como `022`.



Nota

Uma maneira de garantir que a variável `LFS` e o `umask` estejam sempre configuradas corretamente é a de editar o arquivo `.bash_profile` em teu diretório lar pessoal e em `/root/.bash_profile` e inserir os comandos **export** e **umask** acima. Além disso, o shell especificado no arquivo `/etc/passwd` para todos(as) os(as) usantes que precisarem da variável `LFS` precisa ser `bash` para garantir que o arquivo `.bash_profile` seja incorporado como parte do processo de login.

Outra consideração é o método que é usado para logar no sistema anfitrião. Se logar por intermédio de um gerenciador gráfico de exibição, o `.bash_profile` do(a) usante normalmente não é usado quando um terminal virtual for iniciado. Nesse caso, adicione os comandos ao arquivo `.bashrc` para o(a) usante e `root`. Adicionalmente, algumas distribuições usam um teste "if" e não executam as instruções restantes do `.bashrc` para uma invocação não interativa do `bash`. Certifique-se de colocar os comandos antes do teste para uso não interativo.

2.7. Montando a Nova Partição

Agora que um sistema de arquivos tenha sido criado, a partição precisa ser montada, de forma que o sistema anfitrião consiga acessá-la. Este livro presume que o sistema de arquivos esteja montado no diretório especificado pela variável de ambiente `LFS` descrita na seção anterior.

Estritamente falando, ninguém consegue “montar uma partição”. A pessoa monta o *sistema de arquivos* embutido naquela partição. Porém, dado que uma partição não pode conter mais que um sistema de arquivos, as pessoas frequentemente falam da partição e do sistema de arquivos associado como se fossem um e o mesmo.

Crie o ponto de montagem e monte o sistema de arquivos do `LFS` com estes comandos:

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

Substitua `<xxx>` pelo nome da partição do `LFS`.

Se estiver usando múltiplas partições para o `LFS` (por exemplo, uma para `/` e outra para `/home`), [então] monte-as como isto:

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

Substitua `<xxx>` e `<yyy>` pelos nomes apropriados das partições.

Configure o(a) dono(a) e o modo de permissão do diretório `$LFS` (ou seja, o diretório raiz no sistema de arquivos recém-criado para o sistema `LFS`) como `root` e `755` caso a distribuição anfitriã tenha sido configurada para usar um padrão diferente para **mkfs**:

```
chown root:root $LFS
chmod 755 $LFS
```

Assegure-se de que essa nova partição não esteja montada com permissões que sejam restritivas demais (tais como as opções `nosuid` ou `nodelv`). Execute o comando **mount** sem quaisquer parâmetros para ver quais opções estão configuradas para a partição do `LFS` montada. Se `nosuid` e (ou) `nodelv` estiverem configuradas, [então] a partição precisa ser remontada.

Se tiver criado e montado mais partições para o sistema `LFS` que aquela montada em `$LFS`, agora você deveria ajustar a titularidade de propriedade e modo de permissões delas e verificar as opções de montagem também.



Atenção

As instruções acima assumem que você não estará reiniciando seu computador no decorrer do processo do LFS. Se você desligar seu sistema, [então] você ou precisará remontar a partição do LFS a cada vez que você reiniciar o processo de construção, ou modificar o arquivo `/etc/fstab` do sistema anfitrião para remontá-la automaticamente quando você reinicializar. Por exemplo, você poderia acrescentar esta linha ao seu arquivo `/etc/fstab`:

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

Se você usar partições adicionais opcionais, [então] certifique-se de acrescentá-las também.

Se você estiver usando uma partição swap, [então] assegure-se de que ela está habilitada usando o comando **swapon**:

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

Substitua `<zzz>` pelo nome da partição swap.

Agora que a nova partição do LFS está aberta para negócios, é tempo de baixar os pacotes.

Capítulo 3. Pacotes e Remendos

3.1. Introdução

Este capítulo inclui uma lista de pacotes que precisam ser baixados para a finalidade de construir um sistema Linux básico. Os números de versão listados correspondem a versões dos aplicativos que são conhecidos por funcionar e este livro é baseado no uso deles. Nós recomendamos veementemente contra o uso de versões diferentes, pois os comandos de construção para uma versão possivelmente não funcionem com uma versão diferente, a menos que a versão diferente seja especificada por uma errata do LFS ou conselho de segurança. As versões mais novas de pacote possivelmente também tenham problemas que exijam contornos. Esses contornos serão desenvolvidos e estabilizados na versão de desenvolvimento do livro.

Para alguns pacotes, o tarball de lançamento e o tarball instantâneo de repositório (Git ou SVN) para aquele lançamento possivelmente seja publicado com nomes de arquivo semelhantes ou até mesmo idênticos. Porém, o tarball de lançamento possivelmente contenha alguns arquivos que sejam essenciais, embora não armazenados no repositório (por exemplo, um conjunto de comandos sequenciais **configure** gerado pelo **autoconf**), em adição ao conteúdo do correspondente instantâneo de repositório. O livro usa tarballs de lançamento sempre que possível. Usar um instantâneo de repositório em vez de um tarball de lançamento especificado pelo livro causará problemas.

Os locais das transferências possivelmente não estejam sempre acessíveis. Se um local de transferência tiver mudado desde quando este livro foi publicado, [então] o Google (<http://www.google.com/>) fornece um motor de busca útil para a maioria dos pacotes. Se essa busca for mal sucedida, [então] tente um dos meios alternativos de transferência em <https://www.linuxfromscratch.org/lfs/mirrors.html#files>.



Importante

Listados na próxima página estão vários pacotes importantes localizados em `ftpmirror.gnu.org`. O local canônico dos pacotes em foco é `ftp.gnu.org`, mas, devido a um ataque distribuído de negação de serviços (DDOS) de longa duração, o(a) administrador(a) do sítio sugeriu que os(as) editores(as) do LFS usassem `ftpmirror.gnu.org`. Vejam-se *Notícias do Slashdot* para detalhes.

O URL `ftpmirror.gnu.org`, na verdade, redireciona para um dos espelhos do sítio `ftp.gnu.org`. Você também pode selecionar um espelho manualmente em vez de usar `ftpmirror.gnu.org`. Uma lista dos espelhos está localizada em <https://www.gnu.org/prep/ftp.en.html>. Se você escolher usar a lista wget descrita abaixo, esse arquivo também pode ser modificado para usar teu espelho desejado.

Os pacotes e os remendos baixados precisarão ser armazenados em algum lugar que esteja convenientemente disponível durante a construção inteira. Um diretório de trabalho também é exigido para desempacotar os fontes e construí-los. `$LFS/sources` pode ser usado, tanto como o lugar para armazenar os tarballs e os remendos, quanto como um diretório de trabalho. Usando esse diretório, os elementos exigidos estarão localizados na partição do LFS e estarão disponíveis durante todos os estágios do processo de construção.

Para criar esse diretório, execute o seguinte comando, como usante `root`, antes de começar a sessão de transferência:

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

Torne esse diretório escrevível e sticky. “Sticky” significa que, mesmo se múltiplos(as) usantes tiverem permissão de escrita sobre um diretório, somente o(a) dono(a) de um arquivo conseguirá deletar o arquivo dentro de um diretório sticky. O seguinte comando habilitará os modos escrita e sticky:

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

Existem muitas maneiras para obter todos os pacotes e remendos necessários para construir o LFS:

- Os arquivos podem ser transferidos individualmente conforme descrito nas próximas duas seções.

- Para versões estáveis do livro, um tarball de todos os arquivos necessários pode ser transferido a partir de um dos sítios espelhos listados em <https://www.linuxfromscratch.org/mirrors.html#files>.
- Os arquivos podem ser transferidos usando o **wget** e uma lista wget conforme descrito abaixo.

Para transferir todos os pacotes e os remendos usando a *wget-list-systemd* como uma entrada gerada para o comando **wget**, use:

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

Adicionalmente, começando com o LFS-7.0, existe um arquivo separado, *md5sums*, que pode ser usado para verificar se todos os pacotes corretos estão disponíveis antes de prosseguir. Coloque aquele arquivo em *\$LFS/sources* e execute:

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

Essa verificação pode ser usada após recuperar os arquivos necessários com qualquer dos métodos listados acima.

Se os pacotes e os remendos forem transferidos como um(a) usante não *root*, esses arquivos serão de titularidade de propriedade do(a) usante. O sistema de arquivos registra o(a) proprietário(a) pelo UID dele(a) e o UID de um(a) usante normal na distribuição anfitriã não é atribuído no LFS. Assim, os arquivos serão deixados de propriedade de um UID sem nome no sistema LFS final. Se você não atribuirá o mesmo UID para teu(ua) usante no sistema LFS, mude os(as) proprietários(as) desses arquivos para *root* agora para evitar esse problema:

```
chown root:root $LFS/sources/*
```

3.2. Todos os Pacotes



Nota

Leia os *avisos de segurança* antes de transferir os pacotes para descobrir se uma versão mais nova de qualquer pacote deveria ser usada para evitar vulnerabilidades de segurança.

Os fontes do(a) desenvolvedor(a) possivelmente removam lançamentos antigos, especialmente quando esses lançamentos contenham uma vulnerabilidade de segurança. Se um URL abaixo não estiver alcançável, [então] você deveria ler os avisos de segurança primeiro para descobrir se uma versão mais nova (com a vulnerabilidade consertada) deveria ser usada. Se não, [então] tente transferir o pacote removido a partir de um espelho. Apesar de ser possível transferir um lançamento antigo a partir de um espelho, mesmo se esse lançamento tiver sido removido por causa de uma vulnerabilidade, não é uma boa ideia usar um lançamento conhecido por ser vulnerável quando da construção do seu sistema.

Transfira ou de outra forma obtenha os seguintes pacotes:

• **Acl (2.4.0) - 376 KB:**

Página inicial: <https://savannah.nongnu.org/projects/acl>

Transferência: <https://download.savannah.gnu.org/releases/acl/acl-2.4.0.tar.xz>

Soma de verificação MD5: e289f370161698a96f50b2a3fdadf411

• **Attr (2.6.0) - 495 KB:**

Página inicial: <https://savannah.nongnu.org/projects/attr>

Transferência: <https://download.savannah.gnu.org/releases/attr/attr-2.6.0.tar.gz>

Soma de verificação MD5: 910a09c3ae0a211569d180586371cf7e

• **Autoconf (2.73) - 1.385 KB:**

Página inicial: <https://www.gnu.org/software/autoconf/>

Transferência: <https://ftpmirror.gnu.org/autoconf/autoconf-2.73.tar.xz>

Soma de verificação MD5: 9cbad9a116ef845bafed8e7bc771bf4

• Automake (1.18.1) - 1.614 KB:

Página inicial: <https://www.gnu.org/software/automake/>

Transferência: <https://ftpmirror.gnu.org/automake/automake-1.18.1.tar.xz>

Soma de verificação MD5: cea31dbf1120f890cbf2a3032cfb9a68

• Bash (5.3) - 11.089 KB:

Página inicial: <https://www.gnu.org/software/bash/>

Transferência: <https://ftpmirror.gnu.org/bash/bash-5.3.tar.gz>

Soma de verificação MD5: 977c8c0c5ae6309191e7768e28ebc951

• Bc (7.0.3) - 464 KB:

Página inicial: <https://github.com/gavinhoward>

Transferência: <https://github.com/gavinhoward/bc/releases/download/7.0.3/bc-7.0.3.tar.xz>

Soma de verificação MD5: ad4db5a0eb4fdbb3f6813be4b6b3da74

• Binutils (2.47) - 28.355 KB:

Página inicial: <https://www.gnu.org/software/binutils/>

Transferência: <https://sourceware.org/pub/binutils/releases/binutils-2.47.tar.xz>

Soma de verificação MD5: d772acfbfd55a81644e9fe7b8189cd2a5

• Bison (3.8.2) - 2.752 KB:

Página inicial: <https://www.gnu.org/software/bison/>

Transferência: <https://ftpmirror.gnu.org/bison/bison-3.8.2.tar.xz>

Soma de verificação MD5: c28f119f405a2304ff0a7ccdcc629713

• Bzip2 (1.0.8) - 792 KB:

Transferência: <https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz>

Soma de verificação MD5: 67e051268d0c475ea773822f7500d0e5

• Coreutils (9.11) - 6.409 KB:

Página inicial: <https://www.gnu.org/software/coreutils/>

Transferência: <https://ftpmirror.gnu.org/coreutils/coreutils-9.11.tar.xz>

Soma de verificação MD5: e52e9857e4aa9ae38ef32f8ed6a27604

• D-Bus (1.16.2) - 1.090 KB:

Página inicial: <https://www.freedesktop.org/wiki/Software/dbus>

Transferência: <https://dbus.freedesktop.org/releases/dbus/dbus-1.16.2.tar.xz>

Soma de verificação MD5: 97832e6f0a260936d28536e5349c22e5

• DejaGNU (1.6.3) - 608 KB:

Página inicial: <https://www.gnu.org/software/dejagnu/>

Transferência: <https://ftpmirror.gnu.org/dejagnu/dejagnu-1.6.3.tar.gz>

Soma de verificação MD5: 68c5208c58236eba447d7d6d1326b821

• Diffutils (3.12) - 1.894 KB:

Página inicial: <https://www.gnu.org/software/diffutils/>

Transferência: <https://ftpmirror.gnu.org/diffutils/diffutils-3.12.tar.xz>

Soma de verificação MD5: d1b18b20868fb561f77861cd90b05de4

• E2fsprogs (1.47.4) - 9.870 KB:

Página inicial: <https://e2fsprogs.sourceforge.net/>

Transferência: <https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.4/e2fsprogs-1.47.4.tar.gz>

Soma de verificação MD5: 733a93bc688314834ff2d10530ff4dc4

• Elfutils (0.195) - 11.751 KB:

Página inicial: <https://sourceware.org/elfutils/>

Transferência: <https://sourceware.org/ftp/elfutils/0.195/elfutils-0.195.tar.bz2>

Soma de verificação MD5: 3e8ef51c43beceddbbb74879549cf72c

• Expat (2.8.3) - 506 KB:

Página inicial: <https://libexpat.github.io/>

Transferência: https://github.com/libexpat/libexpat/releases/download/R_2_8_3/expat-2.8.3.tar.xz

Soma de verificação MD5: 1747f5a57b191f0373baeb5adc8078ca

• Expect (5.45.4) - 618 KB:

Página inicial: <https://core.tcl.tk/expect/>

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

Soma de verificação MD5: 00fce8de158422f5ccd2666512329bd2

• File (5.48) - 2.631 KB:

Página inicial: <https://www.darwinsys.com/file/>

Transferência: <https://astron.com/pub/file/file-5.48.tar.gz>

Soma de verificação MD5: 423686e97f731d8c24e9cd1a22b03dec

• Findutils (4.11.0) - 2.394 KB:

Página inicial: <https://www.gnu.org/software/findutils/>

Transferência: <https://ftpmirror.gnu.org/findutils/findutils-4.11.0.tar.xz>

Soma de verificação MD5: 512c6875ed84034dca240c4bc9380b96

• Flex (2.6.4) - 1.386 KB:

Página inicial: <https://github.com/westes/flex>

Transferência: <https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz>

Soma de verificação MD5: 2882e3179748cc9f9c23ec593d6adc8d

• Flit-core (4.0.2) - 52 KB:

Página inicial: <https://pypi.org/project/flit-core/>

Transferência: https://pypi.org/packages/source/f/flit-core/flit_core-4.0.2.tar.gz

Soma de verificação MD5: 89ebb783230fc8db6c1d2a39644b7085

• Gawk (5.4.1) - 3.749 KB:

Página inicial: <https://www.gnu.org/software/gawk/>

Transferência: <https://ftpmirror.gnu.org/gawk/gawk-5.4.1.tar.xz>

Soma de verificação MD5: d379c2110e7a3e15347dd5559a41ad64

• GCC (16.2.0) - 104.689 KB:

Página inicial: <https://gcc.gnu.org/>

Transferência: <https://ftpmirror.gnu.org/gcc/gcc-16.2.0/gcc-16.2.0.tar.xz>

Soma de verificação MD5: 19b777fb19ea4982731392481306f0d3

• GDBM (1.26) - 1.198 KB:

Página inicial: <https://www.gnu.org/software/gdbm/>

Transferência: <https://ftpmirror.gnu.org/gdbm/gdbm-1.26.tar.gz>

Soma de verificação MD5: aaa600665bc89e2febb3c7bd90679115

• Gettext (1.0) - 10.471 KB:

Página inicial: <https://www.gnu.org/software/gettext/>

Transferência: <https://ftpmirror.gnu.org/gettext/gettext-1.0.tar.xz>

Soma de verificação MD5: dc8b2911535929cec1e263706b0a13a1

• Glibc (2.44) - 20.138 KB:

Página inicial: <https://www.gnu.org/software/libc/>

Transferência: <https://ftpmirror.gnu.org/glibc/glibc-2.44.tar.xz>

Soma de verificação MD5: 7677da43ef759c68e005f5d4c37986a6

**Nota**

Os(As) desenvolvedores(as) da Glibc mantém uma *ramificação Git* contendo remendos considerados valiosos para a Glibc-2.44, porém infelizmente desenvolvidos depois do lançamento da Glibc-2.44. Os(As) editores(as) do LFS emitirão um aviso de segurança se alguma correção de segurança for acrescentada na ramificação, porém nenhuma ação será tomada para outros remendos acrescentados recentemente. Você possivelmente reveja os remendos você mesmo(a) e incorpore alguns remendos se você os considerar importantes.

• GMP (6.3.0) - 2.046 KB:

Página inicial: <https://www.gnu.org/software/gmp/>

Transferência: <https://ftpmirror.gnu.org/gmp/gmp-6.3.0.tar.xz>

Soma de verificação MD5: 956dc04e864001a9c22429f761f2c283

• Gperf (3.3) - 1.789 KB:

Página inicial: <https://www.gnu.org/software/gperf/>

Transferência: <https://ftpmirror.gnu.org/gperf/gperf-3.3.tar.gz>

Soma de verificação MD5: 31753b021ea78a21f154bf9eeeb8b079

• Grep (3.12) - 1.874 KB:

Página inicial: <https://www.gnu.org/software/grep/>

Transferência: <https://ftpmirror.gnu.org/grep/grep-3.12.tar.xz>

Soma de verificação MD5: 5d9301ed9d209c4a88c8d3a6fd08b9ac

• Groff (1.24.1) - 8.795 KB:

Página inicial: <https://www.gnu.org/software/groff/>

Transferência: <https://ftpmirror.gnu.org/groff/groff-1.24.1.tar.gz>

Soma de verificação MD5: d9d996af3d05ebab934380be708ea584

• GRUB (2.14) - 7.545 KB:

Página inicial: <https://www.gnu.org/software/grub/>

Transferência: <https://ftpmirror.gnu.org/grub/grub-2.14.tar.xz>

Soma de verificação MD5: 383f9effad01c235d2535357ff717543

• Gzip (1.14) - 865 KB:

Página inicial: <https://www.gnu.org/software/gzip/>

Transferência: <https://ftpmirror.gnu.org/gzip/gzip-1.14.tar.xz>

Soma de verificação MD5: 4bf5a10f287501ee8e8e8e00ef62b2c2

• Iana-Etc (20260805) - 596 KB:

Página inicial: <https://www.iana.org/protocols>

Transferência: <https://github.com/Mic92/iana-etc/releases/download/20260805/iana-etc-20260805.tar.gz>

Soma de verificação MD5: f6d2619cfac3a68f91392e28fe4067c3

• Inetutils (2.8) - 2.916 KB:

Página inicial: <https://www.gnu.org/software/inetutils/>

Transferência: <https://ftpmirror.gnu.org/inetutils/inetutils-2.8.tar.gz>

Soma de verificação MD5: f0f9ad13045a67da9c882148f9efae6b

- **IPRoute2 (7.1.0) - 943 KB:**

Página inicial: <https://www.kernel.org/pub/linux/utils/net/iproute2/>

Transferência: <https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-7.1.0.tar.xz>

Soma de verificação MD5: 4f322144fb9a0877c5f9ef71b2f11180

- **Jinja2 (3.1.6) - 240 KB:**

Página inicial: <https://jinja.palletsprojects.com/en/3.1.x/>

Transferência: <https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.6.tar.gz>

Soma de verificação MD5: 66d4c25ff43d1deaf9637ccda523dec8

- **Kbd (2.10.0) - 1.747 KB:**

Página inicial: <https://kbd-project.org/>

Transferência: <https://www.kernel.org/pub/linux/utils/kbd/kbd-2.10.0.tar.xz>

Soma de verificação MD5: 77ff950dfa4c34fdf41125547bfa831a

- **Kmod (34.2) - 434 KB:**

Página inicial: <https://github.com/kmod-project/kmod>

Transferência: <https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-34.2.tar.xz>

Soma de verificação MD5: 36f2cc483745e81ede3406fa55e1065a

- **Less (704) - 974 KB:**

Página inicial: <https://www.greenwoodsoftware.com/less/>

Transferência: <https://www.greenwoodsoftware.com/less/less-704.tar.gz>

Soma de verificação MD5: c4c9eea5b1862783b078c67ca67c9185

- **Libcap (2.78) - 197 KB:**

Página inicial: <https://sites.google.com/site/fullycapable/>

Transferência: <https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.78.tar.xz>

Soma de verificação MD5: 0ff11419aa4813c0a0f84fe67bf3b39e

- **Libffi (3.8.0) - 1.545 KB:**

Página inicial: <https://sourceware.org/libffi/>

Transferência: <https://github.com/libffi/libffi/releases/download/v3.8.0/libffi-3.8.0.tar.gz>

Soma de verificação MD5: 96fe46f2a7bb82559b35a268a2c000ba

- **Libpipeline (1.5.8) - 1.046 KB:**

Página inicial: <https://libpipeline.nongnu.org/>

Transferência: <https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.8.tar.gz>

Soma de verificação MD5: 17ac6969b2015386bcb5d278a08a40b5

- **Libtool (2.6.2) - 1.068 KB:**

Página inicial: <https://www.gnu.org/software/libtool/>

Transferência: <https://ftpmirror.gnu.org/libtool/libtool-2.6.2.tar.xz>

Soma de verificação MD5: 87743862e6cabfb9e76651bc64e31e5c

- **Libxcrypt (4.5.2) - 655 KB:**

Página inicial: <https://github.com/besser82/libxcrypt/>

Transferência: <https://github.com/besser82/libxcrypt/releases/download/v4.5.2/libxcrypt-4.5.2.tar.xz>

Soma de verificação MD5: 25e888919ddcd153a07daa95224fa436

• **Linux (7.1.8) - 154.722 KB:**

Página inicial: <https://www.kernel.org/>

Transferência: <https://www.kernel.org/pub/linux/kernel/v7.x/linux-7.1.8.tar.xz>

Soma de verificação MD5: 807584f564568b40162e9e4a72bf191f



Nota

O kernel do Linux é atualizado com bastante frequência, muitas vezes devido a descobertas de vulnerabilidades de segurança. A mais recente versão estável do kernel disponível pode ser usada, a menos que a página da errata diga o contrário.

Para usantes com largura de banda de velocidade limitada ou cara que desejem atualizar o núcleo Linux, uma versão da linha base do pacote e dos remendos pode ser transferida separadamente. Isso possivelmente economize algum tempo ou custo para uma posterior atualização de nível de remendo contendo um lançamento secundário.

Como exemplo, para linux-7.1.8, o seguinte poderia ser baixado:

- <https://www.kernel.org/pub/linux/kernel/v6.x/linux-7.1.tar.xz>
- <https://www.kernel.org/pub/linux/kernel/v6.x/patch-7.1.8.xz>

Então na Seção 5.4, “Cabeçalhos da API do Linux-7.1.8” e na Seção 10.3, “Linux-7.1.8” desempacote o núcleo, mude para o diretório do pacote e então aplique o remendo com `xzcat ../patch-7.1.8.xz | patch -Np1`.

Neste ponto, se uma versão pontual mais recente do núcleo for necessária, então apenas o remendo mais recente será necessário. No entanto, se uma nova versão secundária for desejada, então tanto a versão secundária completa quanto o remendo desejado precisarão ser baixados.

• **Lz4 (1.10.0) - 379 KB:**

Página inicial: <https://lz4.org/>

Transferência: <https://github.com/lz4/lz4/releases/download/v1.10.0/lz4-1.10.0.tar.gz>

Soma de verificação MD5: dead9f5f1966d9ae56e1e32761e4e675

• **M4 (1.4.21) - 2.032 KB:**

Página inicial: <https://www.gnu.org/software/m4/>

Transferência: <https://ftpmirror.gnu.org/m4/m4-1.4.21.tar.xz>

Soma de verificação MD5: 8051eef7239b2f187791f2ab0034d6b7

• **Make (4.4.1) - 2.300 KB:**

Página inicial: <https://www.gnu.org/software/make/>

Transferência: <https://ftpmirror.gnu.org/make/make-4.4.1.tar.gz>

Soma de verificação MD5: c8469a3713cbb04d955d4ae4be23eeb

• **Man-DB (2.13.1) - 2.061 KB:**

Página inicial: <https://www.nongnu.org/man-db/>

Transferência: <https://download.savannah.gnu.org/releases/man-db/man-db-2.13.1.tar.xz>

Soma de verificação MD5: b6335533cbeac3b24cd7be31fdee8c83

• **Man-pages (6.18) - 1.859 KB:**

Página inicial: <https://www.kernel.org/doc/man-pages/>

Transferência: <https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.18.tar.xz>

Soma de verificação MD5: 44915a91e4a28161dc421636a193fb3a

• **MarkupSafe (3.0.3) - 79 KB:**

Página inicial: <https://palletsprojects.com/p/markupsafe/>

Transferência: <https://pypi.org/packages/source/M/MarkupSafe/markupsafe-3.0.3.tar.gz>

Soma de verificação MD5: 13a73126d25afa72a1ff0daed072f5fe

• Meson (1.12.0) - 2.460 KB:

Página inicial: <https://mesonbuild.com>

Transferência: <https://github.com/mesonbuild/meson/releases/download/1.12.0/meson-1.12.0.tar.gz>

Soma de verificação MD5: 9f943c267d63016012efcb0f31ecdd7d

• MPC (1.4.1) - 520 KB:

Página inicial: <https://www.multiprecision.org/>

Transferência: <https://ftpmirror.gnu.org/mpc/mpc-1.4.1.tar.xz>

Soma de verificação MD5: 40c5fba305125279ea46041160e81daf

• mpdecimal (4.0.1) - 309 KB:

Página inicial: <https://www.bytereef.org/mpdecimal/>

Transferência: <https://www.bytereef.org/software/mpdecimal/releases/mpdecimal-4.0.1.tar.gz>

Soma de verificação MD5: 3d4412579b1edc1047efab151889cca9

• MPFR (4.2.2) - 1.471 KB:

Página inicial: <https://www.mpfr.org/>

Transferência: <https://ftpmirror.gnu.org/mpfr/mpfr-4.2.2.tar.xz>

Soma de verificação MD5: 7c32c39b8b6e3ae85f25156228156061

• Ncurses (6.6) - 3.703 KB:

Página inicial: <https://www.gnu.org/software/ncurses/>

Transferência: <https://invisible-mirror.net/archives/ncurses/ncurses-6.6.tar.gz>

Soma de verificação MD5: dd45bf6854430af403452a7a6a40652c

• Ninja (1.13.2) - 286 KB:

Página inicial: <https://ninja-build.org/>

Transferência: <https://github.com/ninja-build/ninja/archive/v1.13.2/ninja-1.13.2.tar.gz>

Soma de verificação MD5: 76c00637fde44909cd7d56f8d73f2042

• OpenSSL (4.0.1) - 53.789 KB:

Página inicial: <https://www.openssl-library.org/>

Transferência: <https://github.com/openssl/openssl/releases/download/openssl-4.0.1/openssl-4.0.1.tar.gz>

Soma de verificação MD5: 07e316afe26b61e72206b81706b497bb

• Packaging (26.3) - 307 KB:

Página inicial: <https://pypi.org/project/packaging/>

Transferência: <https://files.pythonhosted.org/packages/source/p/packaging/packaging-26.3.tar.gz>

Soma de verificação MD5: c6809ec56986682ceffe573b8778a1ce

• Patch (2.8) - 886 KB:

Página inicial: <https://savannah.gnu.org/projects/patch/>

Transferência: <https://ftpmirror.gnu.org/patch/patch-2.8.tar.xz>

Soma de verificação MD5: 149327a021d41c8f88d034eab41c039f

• Pcre2 (10.47) - 2.096 KB:

Página inicial: <https://github.com/PCRE2Project/pcre2/>

Transferência: <https://github.com/PCRE2Project/pcre2/releases/download/pcre2-10.47/pcre2-10.47.tar.bz2>

Soma de verificação MD5: aded5840ab5a7d772dd4e16fc294b665

• Perl (5.44.0) - 14.571 KB:

Página inicial: <https://www.perl.org/>

Transferência: <https://www.cpan.org/src/5.0/perl-5.44.0.tar.xz>

Soma de verificação MD5: 55761cf1543af326492fd194647796d1

• Pkgconf (3.0.5) - 417 KB:

Página inicial: <https://github.com/pkgconf/pkgconf>

Transferência: <https://distfiles.ariadne.space/pkgconf/pkgconf-3.0.5.tar.xz>

Soma de verificação MD5: 18c528161e6b04ae36dc981af8c4d97e

• Procps (4.0.7) - 1.552 KB:

Página inicial: <https://gitlab.com/procps-ng/procps/>

Transferência: <https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.7.tar.xz>

Soma de verificação MD5: d47bdda51c4bd595a1ab21b8e538bcb3

• Psmisc (23.7) - 423 KB:

Página inicial: <https://gitlab.com/psmisc/psmisc>

Transferência: <https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.7.tar.xz>

Soma de verificação MD5: 53eae841735189a896d614cba440eb10

• Python (3.14.7) - 23.911 KB:

Página inicial: <https://www.python.org/>

Transferência: <https://www.python.org/ftp/python/3.14.7/Python-3.14.7.tar.xz>

Soma de verificação MD5: a4c4c2991b4126e227073c2d9445f432

• Documentação do Python (3.14.7) - 10.916 KB:

Transferência: <https://www.python.org/ftp/python/doc/3.14.7/python-3.14.7-docs-html.tar.bz2>

Soma de verificação MD5: bdf55dddc02a76961d61730f06d18849

• Readline (8.3) - 3.340 KB:

Página inicial: <https://tiswww.case.edu/php/chet/readline/rltop.html>

Transferência: <https://ftpmirror.gnu.org/readline/readline-8.3.tar.gz>

Soma de verificação MD5: 25a73bfb2a3ad7146c5e9d4408d9f6cd

• Sed (4.10) - 1.693 KB:

Página inicial: <https://www.gnu.org/software/sed/>

Transferência: <https://ftpmirror.gnu.org/sed/sed-4.10.tar.xz>

Soma de verificação MD5: c70dc5372db95c816442ffedf77a0d0f

• Setuptools (84.0.0) - 1.142 KB:

Página inicial: <https://pypi.org/project/setuptools/>

Transferência: <https://pypi.org/packages/source/s/setuptools/setuptools-84.0.0.tar.gz>

Soma de verificação MD5: c6919a4b0862ca30604559e163a6cce9

• Shadow (4.20.2) - 2.238 KB:

Página inicial: <https://github.com/shadow-maint/shadow/>

Transferência: <https://github.com/shadow-maint/shadow/releases/download/4.20.2/shadow-4.20.2.tar.xz>

Soma de verificação MD5: f52e717bd330818c67bd57c897ec5a8e

• Sqlite (3530400) - 3.207 KB:

Página inicial: <https://sqlite.org>

Transferência: <https://sqlite.org/2026/sqlite-autoconf-3530400.tar.gz>

Soma de verificação MD5: ee54db5a5b04e77d4883ce0c051a1eba

• Documentação do Sqlite (3530400) - 11.544 KB:

Página inicial: <https://sqlite.org>

Transferência: <https://sqlite.org/2026/sqlite-doc-3530400.zip>

Soma de verificação MD5: 2cd5b168d0e5c5ae2b99beb3a4269a73

• **Systemd (261.2) - 18.024 KB:**

Página inicial: <https://systemd.io>

Transferência: <https://github.com/systemd/systemd/archive/v261.2/systemd-261.2.tar.gz>

Soma de verificação MD5: e3f04ccc83a28ec5e80c5432bb8329f5

• **Páginas de Manual do Systemd (261.2) - 803 KB:**

Página inicial: <https://systemd.io>

Transferência: <https://andu.in.linuxfromscratch.org/LFS/systemd-man-pages-261.2.tar.xz>

Soma de verificação MD5: 4c5533c1a2d0921dd4c5b5e6b8902570



Nota

A equipe do Linux From Scratch gera o próprio tarball dela das páginas de manual usando o fonte do systemd. Isso é feito com a finalidade de evitar dependências desnecessárias.

• **Tar (1.35) - 2.263 KB:**

Página inicial: <https://www.gnu.org/software/tar/>

Transferência: <https://ftpmirror.gnu.org/tar/tar-1.35.tar.xz>

Soma de verificação MD5: a2d8042658cfd8ea939e6d911eaf4152

• **Tcl (8.6.18) - 11.540 KB:**

Página inicial: <https://tcl.sourceforge.net/>

Transferência: <https://downloads.sourceforge.net/tcl/tcl8.6.18-src.tar.gz>

Soma de verificação MD5: acfe0c9f7d0c626ecf026e834a888da6

• **Documentação do Tcl (8.6.18) - 1.172 KB:**

Transferência: <https://downloads.sourceforge.net/tcl/tcl8.6.18-html.tar.gz>

Soma de verificação MD5: 54d1ff0f5eee4e81e5cdaa4baa343397

• **Texinfo (7.3) - 6.778 KB:**

Página inicial: <https://www.gnu.org/software/texinfo/>

Transferência: <https://ftpmirror.gnu.org/texinfo/texinfo-7.3.tar.xz>

Soma de verificação MD5: 915a09fcd9c2bf0f5ecf9e556d5698ff

• **Dados de Fuso Horário (2026c) - 465 KB:**

Página inicial: <https://www.iana.org/time-zones>

Transferência: <https://www.iana.org/time-zones/repository/releases/tzdata2026c.tar.gz>

Soma de verificação MD5: bff7174205cefab793e3b24271ef2f45

• **Util-linux (2.42.2) - 10.409 KB:**

Página inicial: <https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/>

Transferência: <https://www.kernel.org/pub/linux/utils/util-linux/v2.42/util-linux-2.42.2.tar.xz>

Soma de verificação MD5: 1d70131b70abda3dec3b37e282a20c96

• **Vim (9.2.1025) - 19.662 KB:**

Página inicial: <https://www.vim.org>

Transferência: <https://github.com/vim/vim/archive/v9.2.1025/vim-9.2.1025.tar.gz>

Soma de verificação MD5: 060c9c0d7e2bdb5e5d80daad596cae02



Nota

A versão do vim muda diariamente. Para obter a versão mais recente, vá até <https://github.com/vim/vim/tags>.

- **Wheel (0.48.0) - 65 KB:**

Página inicial: <https://pypi.org/project/wheel/>

Transferência: <https://pypi.org/packages/source/w/wheel/wheel-0.48.0.tar.gz>

Soma de verificação MD5: f668e4885de814378b332ecee04a8a0

- **Xz Utils (5.8.3) - 1.512 KB:**

Página inicial: <https://tukaani.org/xz>

Transferência: <https://github.com//tukaani-project/xz/releases/download/v5.8.3/xz-5.8.3.tar.xz>

Soma de verificação MD5: a02753f34e5546d20213b87f876a0933

- **Zlib (1.3.2) - 1.468 KB:**

Página inicial: <https://zlib.net/>

Transferência: <https://zlib.net/fossils/zlib-1.3.2.tar.gz>

Soma de verificação MD5: a1e6c958597af3c67d162995a342138a

- **Zstd (1.5.7) - 2.378 KB:**

Página inicial: <https://facebook.github.io/zstd/>

Transferência: <https://github.com//facebook/zstd/releases/download/v1.5.7/zstd-1.5.7.tar.gz>

Soma de verificação MD5: 780fc1896922b1bc52a4e90980cdda48

Tamanho total desses pacotes: cerca de NaN Mo

3.3. Remendos Necessários

Adicionalmente aos pacotes, vários remendos também são exigidos. Esses remendos corrigem alguns erros nos pacotes que deveriam ser consertados pelo(a) Mantenedor(a). Os remendos também fazem pequenas modificações para tornar os pacotes mais fáceis de se trabalhar. Os seguintes remendos serão necessários para construir um sistema LFS:

- **Remendo da Documentação do Bzip2 - 1.6 KB:**

Transferência: https://www.linuxfromscratch.org/patches/lfs/13.1/bzip2-1.0.8-install_docs-1.patch

Soma de verificação MD5: 6a5ac7e89b791aae556de0f745916f7f

- **Remendo de Correções de Internacionalização do Coreutils - 67 KB:**

Transferência: <https://www.linuxfromscratch.org/patches/lfs/13.1/coreutils-9.11-i18n-1.patch>

Soma de verificação MD5: 900d64d9936516b68613271c9ebc0059

- **Remendo GCC15 do Expect - 12 KB:**

Download: <https://www.linuxfromscratch.org/patches/lfs/13.1/expect-5.45.4-gcc15-1.patch>

Soma de verificação MD5: 0ca4d6bb8d572fbcd13cb36cd34833e

- **Remendo de Correções de Fluxo de Desenvolvimento da Glibc - 172 KB:**

Transferência: https://www.linuxfromscratch.org/patches/lfs/13.1/glibc-2.44-upstream_fixes-1.patch

Soma de verificação MD5: 990574184b25aa3029a2444e81865098

- **Remendo do FHS da Glibc - 2.8 KB:**

Transferência: <https://www.linuxfromscratch.org/patches/lfs/13.1/glibc-fhs-1.patch>

Soma de verificação MD5: 9a5997c3452909b1769918c759eff8a2

- **Remendo de Correção do Backspace/Delete do Kbd - 12 KB:**

Transferência: <https://www.linuxfromscratch.org/patches/lfs/13.1/kbd-2.10.0-backspace-1.patch>

Soma de verificação MD5: f75cca16a38da6caa7d52151f7136895

- **Remendo Openssl4 do Python - 38 KB:**

Transferência: https://www.linuxfromscratch.org/patches/lfs/13.1/Python-3.14.7-openssl_4-1.patch

Soma de verificação MD5: 597d7737df1b4ea4e184c193da523050

- **Remendo de Fluxo de Desenvolvimento do Tar - 4.3 KB:**

Transferência: https://www.linuxfromscratch.org/patches/lfs/13.1/tar-1.35-acl_fix-1.patch

Soma de verificação MD5: dbab49e317105539611866dac5dd54f6

Tamanho total desses remendos: cerca de NaN Mo

Adicionalmente aos remendos exigidos acima, existe um número de remendos opcionais criados pela comunidade do LFS. Esses remendos opcionais solucionam problemas menores ou habilitam funcionalidade que não está habilitada por padrão. Sinta-se à vontade para examinar a base de dados dos remendos localizada em <https://www.linuxfromscratch.org/patches/downloads/> e adquirir quaisquer remendos adicionais para atender às necessidades do seu sistema.

Capítulo 4. Preparações Finais

4.1. Introdução

Neste capítulo, nós realizaremos umas poucas tarefas adicionais para preparar para construir o sistema temporário. Nós criaremos um conjunto de diretórios em `$LFS` (no qual nós instalaremos as ferramentas temporárias); adicionaremos um(a) usante desprivilegiado(a); e criaremos um ambiente apropriado de construção para aquele(a) usante. Nós também explicaremos as unidades de tempo (“UPCs”) que usamos para medir quanto tempo leva para construir pacotes do LFS e forneceremos alguma informação acerca de suítes de teste de pacote.

4.2. Criando um Layout Limitado de Diretório no Sistema de Arquivos do LFS

Nesta seção, nós começamos povoando o sistema de arquivos do LFS com os lugares que constituirão o sistema Linux final. O primeiro passo é o de criar uma hierarquia limitada de diretório, de forma que os aplicativos compilados no Capítulo 6 (bem como a `glibc` e a `libstdc++` no Capítulo 5) possam ser instalados no local final deles. Nós fazemos isso, de forma que aqueles aplicativos temporários sejam sobrescritos quando as versões finais sejam reconstruídas no Capítulo 8.

Crie o layout exigido de diretório emitindo os seguintes comandos como `root`:

```
mkdir -pv $LFS/{etc,var} $LFS/usr/{bin,lib,sbin}

for i in bin lib sbin; do
    ln -sv usr/$i $LFS/$i
done

case $(uname -m) in
    x86_64) mkdir -pv $LFS/lib64 ;;
esac
```

Os aplicativos no Capítulo 6 serão compilados com um compilador cruzado (mais detalhes podem ser encontrados na seção Observações Técnicas do Conjunto de Ferramentas). Esse compilador cruzado será instalado em um diretório especial, para separá-lo de outros aplicativos. Ainda atuando como `root`, crie esse diretório com este comando:

```
mkdir -pv $LFS/tools
```



Nota

Os(As) editores(as) do LFS deliberadamente decidiram não usar um diretório `/usr/lib64`. Vários passos são tomados para se ter certeza de que o conjunto de ferramentas não o usará. Se por qualquer razão esse diretório aparecer (seja porque você cometeu um erro ao seguir as instruções, seja porque você instalou um pacote binário que o criou depois de finalizar o LFS), [então] possivelmente quebre o seu sistema. Você deveria sempre ter certeza de que esse diretório não existe.

4.3. Adicionando o(a) Usante LFS

Enquanto logado(a) como usante `root`, cometer um único erro pode danificar ou destruir um sistema. Portanto, os pacotes nos próximos dois capítulos serão construídos como um(a) usante desprivilegiado(a). Você poderia usar teu próprio nome de usante, mas para tornar mais fácil configurar um ambiente limpo e funcional, nós criaremos um(a) usante novo(a) chamado(a) `lfs` como um(a) membro(a) de um novo grupo (também nominado de `lfs`) e executaremos comandos como `lfs` durante o processo de instalação. Como `root`, emita os seguintes comandos para adicionar o(a) novo(a) usante:

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

Isto é o que as opções da linha de comando significam:

`-s /bin/bash`

Isso torna **bash** o shell padrão para usante `lfs`.

`-g lfs`

Essa opção acrescenta usante `lfs` ao grupo `lfs`.

`-m`

Isso cria um diretório home para `lfs`.

`-k /dev/null`

Esse parâmetro evita possível cópia de arquivos a partir de um diretório esqueleto (o padrão é `/etc/skel`) mudando o local da entrada gerada para o dispositivo especial `null`.

`lfs`

Esse é o nome do(a) novo(a) usante.

Se quiser logar como `lfs` ou alternar para `lfs` a partir de um(a) usante não `root` (em oposição a alternar para o(a) usante `lfs` quando estiver logado(a) como `root`, o que não exige que o(a) usante `lfs` tenha uma senha), você precisa configurar uma senha para `lfs`. Emita o seguinte comando como o(a) usante `root` para configurar a senha:

```
passwd lfs
```

Conceda a `lfs` acesso total a todos os diretórios sob `$LFS` tornando `lfs` a(o) dona(o) do diretório:

```
chown -v lfs $LFS/{usr{,/*},var,etc,tools}
case $(uname -m) in
  x86_64) chown -v lfs $LFS/lib64 ;;
esac
```

**Nota**

Em alguns sistemas anfitriões, o seguinte comando **su** não completa adequadamente e suspende o login para o(a) usante `lfs` para o segundo plano. Se o prompt "`lfs:~$`" não aparecer imediatamente, inserir o comando **fg** corrigirá o problema.

Em seguida, inicie um shell executando como usante `lfs`. Isso pode ser feito logando-se como `lfs` em um console virtual ou com o seguinte comando de substituir/comutar usante:

```
su - lfs
```

O "`-`" instrui "**su**" a iniciar um "shell" de "login" em vez de um "shell" de não "login". A diferença entre esses dois tipos de "shells" está descrita em detalhes em "*bash(1)*" e "**info bash**".

4.4. Configurando o Ambiente

Configure um bom ambiente de trabalho criando dois novos arquivos de inicialização para o shell **bash**. Enquanto logado(a) como usante `lfs`, emita o seguinte comando para criar um novo `.bash_profile`:

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

Enquanto logado(a) como usante `lfs` ou quando comutado para o(a) usante `lfs` usando um comando **su** com a opção "`-`", o shell inicial é um shell de *login* que lê o `/etc/profile` do anfitrião (provavelmente contendo algumas configurações e variáveis de ambiente) e então `.bash_profile`. O comando **exec env -i.../bin/bash** no arquivo `.bash_`

profile substitui o shell executante por um novo com um ambiente completamente vazio, exceto pelas variáveis HOME, TERM e PS1. Isso garante que nenhuma variável de ambiente indesejada e potencialmente danosa oriunda do sistema anfitrião vaze para o ambiente de construção.

A nova instância do shell é um shell de *não-login*, que não lê, e executa, o conteúdo dos arquivos `/etc/profile` ou `.bash_profile`, porém, ao invés, lê, e executa, o arquivo `.bashrc`. Crie o arquivo `.bashrc` agora:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT PATH CONFIG_SITE
EOF
```

O significado das configurações em `.bashrc`

```
set +h
```

O comando **set +h** desliga a função hash do **bash**. “Hashing” geralmente é uma característica útil—**bash** usa uma tabela hash para lembrar o caminho completo para arquivos executáveis para evitar procurar o PATH várias vezes para encontrar o mesmo executável. Entretanto, as novas ferramentas deveriam ser usadas tão logo sejam instaladas. Alternando para desligada a função hash força o shell a procurar no PATH sempre que um aplicativo estiver para ser executado. Dessa forma, o shell encontrará as ferramentas recém compiladas em `$LFS/tools/bin` tão logo elas estejam disponíveis sem lembrar da versão anterior do mesmo aplicativo fornecida pela distribuição anfitriã, em `/usr/bin` ou `/bin`.

```
umask 022
```

Configura o umask conforme nós já explicamos em Seção 2.6, “Configurando a Variável \$LFS e o Umask.”

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

A variável LFS deveria ser configurada para o ponto de montagem escolhido.

```
LC_ALL=POSIX
```

A variável LC_ALL controla a localização de certos aplicativos, fazendo suas mensagens seguirem as convenções de um país especificado. Configurar LC_ALL para “POSIX” ou “C” (as duas são equivalentes) garante que tudo vai funcionar como esperado no ambiente de compilação cruzada.

```
LFS_TGT=$(uname -m)-lfs-linux-gnu
```

A variável LFS_TGT configura uma não padrão, porém compatível descrição de máquina para uso quando da construção do nosso compilador cruzado e vinculador e quando da compilação cruzada do nosso conjunto de ferramentas temporárias. Mais informação é fornecida pelas Observações Técnicas do Conjunto de Ferramentas.

```
PATH=/usr/bin
```

Muitas distribuições modernas do Linux mesclaram `/bin` e `/usr/bin`. Quando esse for o caso, a variável PATH padrão deveria ser configurada para `/usr/bin/` para o ambiente do Capítulo 6. Quando esse não for o caso, a seguinte linha acrescenta `/bin` ao caminho.

```
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
```

Se `/bin` não for um link simbólico, [então] ele precisa ser acrescentado à variável PATH.

```
PATH=$LFS/tools/bin:$PATH
```

Ao se colocar `$LFS/tools/bin` a frente do PATH padrão, o compilador cruzado instalado no início do Capítulo 5 é pego pelo shell imediatamente depois da instalação dele. Isso, combinado com a desativação do hashing, limita o risco de que o compilador originário do anfitrião seja usado em vez do compilador cruzado.

```
CONFIG_SITE=$LFS/usr/share/config.site
```

No Capítulo 5 e no Capítulo 6, se essa variável não estiver configurada, [então] os scripts **configure** possivelmente tentem carregar itens de configuração específicos para algumas distribuições a partir de `/usr/share/config.site` no sistema anfitrião. Substitua-o para evitar uma potencial contaminação oriunda do anfitrião.

```
export ...
```

Ao tempo que os comandos precedentes tenham configurado algumas variáveis, com a finalidade de torná-las visíveis dentro de quaisquer sub-shells, nós as exportamos.



Importante

Muitas distribuições comerciais acrescentam uma instância não documentada de `/etc/bash.bashrc` à inicialização do **bash**. Esse arquivo tem o potencial de modificar o ambiente do(a) usante `lfs` de maneiras que conseguem afetar a construção de pacotes LFS críticos. Para assegurar que o ambiente do(a) usante `lfs` esteja limpo, verifique a presença do `/etc/bash.bashrc` e, se presente, mova-o para fora do caminho. Como o(a) usante `root`, execute:

```
[ ! -e /etc/bash.bashrc ] || mv -v /etc/bash.bashrc /etc/bash.bashrc.NOUSE
```

Quando o(a) usante `lfs` não mais for necessário(a) (no início do Capítulo 7), você pode seguramente restaurar `/etc/bash.bashrc` (se desejado).

Perceba que o pacote Bash do LFS que nós construiremos na Seção 8.39, “Bash-5.3” não é configurado para carregar ou para executar `/etc/bash.bashrc`, de modo que esse arquivo é inútil em um sistema LFS finalizado.

Para muitos sistemas modernos com múltiplos processadores (ou núcleos), o tempo de compilação para um pacote pode ser reduzido executando-se um “make paralelo”, informando-se ao aplicativo “make” quantos processadores estão disponíveis por meio de uma opção de linha de comando ou de uma variável de ambiente. Por exemplo, um processador “Intel” “Core i9-13900K” tem oito (08) núcleos “D” (desempenho) e dezesseis (16) núcleos “E” (eficiência), e um núcleo “D” consegue executar simultaneamente duas camadas, de forma que cada núcleo “D” seja modelado como dois núcleos lógicos pelo núcleo Linux. Como resultado, existem trinta e dois (32) núcleos lógicos no total. Uma maneira óbvia de usar todos esses núcleos lógicos é a de permitir que o “**make**” gere até trinta e duas (32) tarefas de construção. Isso pode ser feito passando-se a opção “-j32” para o “**make**”:

```
make -j32
```

Ou configure a variável de ambiente “MAKEFLAGS” e o conteúdo dela será usado automaticamente por “**make**” como opções de linha de comando:

```
export MAKEFLAGS=-j32
```



Importante

Nunca passe uma opção “-j” sem um número para “**make**” ou configure tal opção em “MAKEFLAGS”. Fazer isso permitirá que “**make**” gere infinitas tarefas de construção e cause problemas de estabilidade do sistema.

Para usar todos os núcleos lógicos disponíveis para construir pacotes em “Capítulo 5” e “Capítulo 6”, configure “MAKEFLAGS” agora em “`.bashrc`”:

```
cat >> ~/.bashrc << "EOF"
export MAKEFLAGS=-j$(nproc)
EOF
```

Substitua “`$(nproc)`” pelo número de núcleos lógicos que você deseja usar se não quiser usar todos os núcleos lógicos.

Finalmente, para garantir que o ambiente esteja totalmente preparado para construir as ferramentas temporárias, force o shell **bash** a ler o perfil do(a) novo(a) usante:

```
source ~/.bash_profile
```

4.5. A Respeito de UPCs

Muitas pessoas gostariam de saber de antemão aproximadamente quanto tempo leva para compilar e instalar cada pacote. Devido ao Linux From Scratch poder ser construído em muitos sistemas, é impossível fornecer estimativas de tempo absolutas. O maior pacote (gcc) levará aproximadamente cinco (05) minutos nos sistemas mais rápidos, mas poderia levar dias nos sistemas mais lentos! Em vez de fornecer tempos atuais, a medida Unidade Padrão de Construção (UPC) será usada.

A medida UPC funciona como segue. O primeiro pacote a ser compilado é o binutils no Capítulo 5. O tempo necessário para compilar usando um núcleo é o que nós nos referiremos como a Unidade Padrão de Construção ou UPC. Todos os outros tempos de compilação serão expressos relativamente a esse tempo.

Por exemplo, considere um pacote cujo tempo de compilação é de quatro e meio (4,5) UPCs. Isso significa que, se o teu sistema precisou de quatro (04) minutos para compilar e instalar a primeira passagem do binutils, será necessário *aproximadamente* dezoito (18) minutos para construir o pacote de exemplo. Felizmente, a maioria dos tempos de construção é mais curta que uma UPC.

As UPCs não são totalmente precisas, pois dependem de muitos fatores, incluindo a versão do GCC do sistema anfitrião. Elas são fornecidas aqui para dar uma estimativa de quanto tempo pode levar para instalar um pacote, mas os números podem variar tanto quanto dúzias de minutos em alguns casos.

Em alguns sistemas mais novos, a placa-mãe é capaz de controlar a velocidade do clock do sistema. Isso pode ser controlado com um comando como **powerprofilesctl**. Isso não está disponível no LFS, mas pode estar disponível na distribuição anfitriã. Depois que o LFS estiver completo, ele pode ser adicionado a um sistema com os procedimentos na página *power-profiles-daemon do BLFS*. Antes de mensurar o tempo de construção de qualquer pacote, é aconselhável usar um perfil de eletricidade do sistema configurado para desempenho máximo (e consumo máximo de eletricidade). Caso contrário, o valor da UPC mensurado possivelmente seja impreciso porque o sistema possivelmente reaja diferentemente ao construir binutils-passagem1 ou outros pacotes. Esteja ciente de que uma imprecisão significativa ainda pode aparecer mesmo se o mesmo perfil for usado para ambos os pacotes, porque o sistema possivelmente responda mais lentamente se estiver ocioso ao iniciar o procedimento de construção. Configurar o perfil de eletricidade para “performance” minimizará esse problema. E, obviamente, fazer isso também fará com que o sistema construa o LFS mais rápido.

Se **powerprofilesctl** estiver disponível, emita o comando **powerprofilesctl set performance** para selecionar o perfil performance. Algumas distribuições fornecem o comando **tuned-adm** para gerenciar os perfis em vez de **powerprofilesctl**; nessas distribuições emita o comando **tuned-adm profile throughput-performance** para selecionar o perfil throughput-performance.



Nota

Quando múltiplos processadores são usados dessa maneira, as unidades UPC no livro variarão ainda mais do que normalmente aconteceria. Em alguns casos, o passo make simplesmente falhará. Analisar a saída gerada do processo de construção também será mais difícil, pois as linhas originárias de diferentes processos estarão intercaladas. Se você tiver um problema com um passo de construção, [então] retorne para uma construção de processador único para analisar adequadamente as mensagens de erro.

Os tempos apresentados aqui para todos os pacotes (exceto binutils-passagem1, que é baseado em um núcleo) são baseados no uso de quatro núcleos (-j4). Os tempos no Capítulo 8 também incluem o tempo para executar os testes de regressão para o pacote, a menos que especificado de outra forma.

4.6. A Respeito das Suítes de Teste

A maioria dos pacotes fornece uma suíte de teste. Executar a suíte de teste para um pacote recém construído é uma boa ideia, pois pode fornecer uma “verificação de sanidade” indicando que tudo compilou corretamente. Uma suíte de teste que ultrapassa o conjunto de verificações dela geralmente prova que o pacote está funcionando como a(o) desenvolvedora(r) pretendia. Não garante, entretanto, que o pacote está totalmente livre de defeitos.

Algumas suítes de teste são mais importantes que outras. Por exemplo, as suítes de teste para o conjunto de ferramentas central—GCC, binutils e glibc—são de máxima importância devido ao papel central delas em um sistema que funcione adequadamente. As suítes de teste para GCC e glibc podem levar bastante tempo para completarem, especialmente em hardware mais lento, mas são fortemente recomendadas.



Nota

Executar as suítes de teste no Capítulo 5 e no Capítulo 6 é impossível; dado que os aplicativos de teste são compilados com um compilador cruzado, eles provavelmente não conseguem executar no anfitrião de construção.

Um problema comum com a execução de suítes de teste para o binutils e o GCC é ficar sem os pseudo terminais (PTYs). Isso pode resultar em um alto número de testes com falhas. Isso pode acontecer por muitas razões, mas a causa mais provável é a de que o sistema anfitrião não tem o sistema de arquivos devpts configurado corretamente. Esse problema é discutido em maiores detalhes em <https://www.linuxfromscratch.org/lfs/faq.html#no-ptys>.

Algumas vezes suítes de testes de pacotes falharão por razões as quais as(os) desenvolvedoras(es) estão cientes e consideraram não-críticas. Consulte os registros localizados em <https://www.linuxfromscratch.org/lfs/build-logs/13.1/> para verificar quando essas falhas são esperadas ou não. Esse sítio é válido para todas as suítes de teste ao longo deste livro.

Parte III. Construindo o Conjunto de Ferramentas Cruzadas do LFS e Ferramentas Temporárias

Material Preliminar Importante

Introdução

Esta parte é dividida em três estágios: primeiro construir um compilador cruzado e suas bibliotecas associadas; segundo, usar esse conjunto de ferramentas cruzado para construir vários utilitários de uma forma que os isole da distribuição anfitriã; terceiro, entrar no ambiente chroot (o qual melhora ainda mais o isolamento do anfitrião) e construir as ferramentas restantes necessárias para construir o sistema final.



Importante

Essa é onde o trabalho real de construir um novo sistema inicia. Seja muito cuidadoso(a) em seguir as instruções exatamente conforme o livro as mostra. Você deveria tentar entender o que cada comando faz e, não importa o quão ansioso(a) você estiver para finalizar sua construção, você deveria evitar digitar cegamente os comandos como mostrado. Leia a documentação quando houver algo que você não entenda. Além disso, mantenha um rastreio da sua digitação e da saída gerada de comandos, usando o utilitário **tee** para enviar a saída gerada pelo terminal para um arquivo. Isso torna a depuração mais fácil se algo der errado.

A próxima seção é uma introdução técnica ao processo de construção, enquanto que a seguinte apresenta instruções gerais **muito importantes**.

Observações Técnicas do Conjunto de Ferramentas

Esta seção explica algumas das razões e detalhes técnicos por trás do método completo de construção. Não tente imediatamente entender tudo nesta seção. A maior parte desta informação ficará mais clara depois de realizar uma construção real. Volte e releia este capítulo a qualquer tempo durante o processo de construção.

O objetivo geral do Capítulo 5 e do Capítulo 6 é o de produzir uma área temporária contendo um conjunto de ferramentas que sejam conhecidas por serem boas e que estejam isoladas do sistema anfitrião. Usando-se o comando **chroot**, as compilações nos capítulos subsequentes estarão isoladas naquele ambiente, assegurando uma construção limpa e livre de problemas do sistema LFS alvo. O processo de construção foi projetado para minimizar os riscos para novos(as) leitores(as) e para fornecer o maior valor educacional ao mesmo tempo.

O processo de construção é baseado em *compilação cruzada*. Compilação cruzada normalmente é usada para construir um compilador e o conjunto de ferramentas associadas dele para uma máquina diferente daquela que é usada para a construção. Isso não é estritamente necessário para o LFS, pois a máquina onde o novo sistema executará é a mesma que aquela usada para a construção. Porém, a compilação cruzada tem a grande vantagem: tudo o que for compilado cruzadamente não pode depender do ambiente do anfitrião.

Acerca da Compilação Cruzada



Nota

O livro LFS não é (e não contém) um tutorial geral para construir um conjunto cruzado de ferramentas (ou nativo). Não use os comandos no livro para um conjunto cruzado de ferramentas para algum outro propósito que não o de construir o LFS, a menos que você realmente entenda o que está fazendo.

É sabido que instalar o GCC passagem 2 quebrará o conjunto cruzado de ferramentas. Nós não consideramos isso um defeito porque o GCC passagem 2 é o último pacote a ser compilado cruzadamente no livro, e não o “consertaremos” até que realmente precisemos compilar cruzadamente algum pacote depois do GCC passagem 2 no futuro.

Compilação cruzada envolve alguns conceitos que merecem uma seção por si próprios. Embora esta seção possivelmente seja omitida em uma primeira leitura, retornar até ela posteriormente te ajudará a ganhar um entendimento mais completo do processo.

Permita-nos primeiro definir alguns termos usados nesse contexto.

A construtora

é a máquina onde nós construímos programas. Observe que essa máquina também é referenciada como a “anfitriã.”

A anfitriã

é a máquina/sistema onde os programas construídos executarão. Observe que esse uso de “host” não é o mesmo que em outras seções.

O alvo

é usado somente para compiladores. Ele é a máquina para a qual o compilador produz código. Ele possivelmente seja diferente tanto da construtora quanto da anfitriã.

Como um exemplo, permita-nos imaginar o seguinte cenário (ocasionalmente referenciado como “Cruzado Canadense”). Nós temos um compilador somente em uma máquina lenta, vamos chamá-la de máquina A, e o compilador de ccA. Nós também temos uma máquina rápida (B), porém nenhum compilador para (B), e nós queremos produzir código para uma terceira, máquina lenta (C). Nós construiremos um compilador para a máquina C em três estágios.

Estágio	Construtora	Anfitriã	Alvo	Ação
1	A	A	B	Construir compilador cruzado cc1 usando ccA na máquina A.
2	A	B	C	Construir compilador cruzado cc2 usando cc1 na máquina A.
3	B	C	C	Construir compilador ccC usando cc2 na máquina B.

Então, todos os programas necessários para a máquina C podem ser compilados usando cc2 na rápida máquina B. Observe que a menos que B consiga executar programas produzidos por C, não existe maneira de testar os programas recém construídos até que a própria máquina C esteja executando. Por exemplo, para executar uma suíte de teste em ccC, nós possivelmente queiramos adicionar um quarto estágio:

Estágio	Construtora	Anfitriã	Alvo	Ação
4	C	C	C	Reconstruir e testar ccC usando ccC na máquina C.

No exemplo acima, somente cc1 e cc2 são compiladores cruzados, isto é, eles produzem código para uma máquina diferente daquela na qual estão executando. Os outros compiladores, ccA e ccC, produzem código para a máquina na qual estão executando. Tais compiladores são chamados de compiladores *nativos*.

Implementação da Compilação Cruzada para o LFS



Nota

Todos os pacotes compilados cruzadamente neste livro usam um sistema de construção baseado no autoconf. O sistema de construção baseado no autoconf aceita tipos de sistema na forma `cpu-vendor-kernel-os`, referenciado como o *tripleto* do sistema. Dado que o campo *vendor* frequentemente é irrelevante, o autoconf te permite omiti-lo.

Um(a) leitor(a) astuto(a) possivelmente questione porque um “tripleto” se refere a um nome de quatro componentes. O campo *kernel* e o campo *os* iniciaram como um campo único do “sistema”. Tal forma de três campos ainda é válida atualmente para alguns sistemas, por exemplo, `x86_64-unknown-freebsd`. Porém, dois sistemas conseguem compartilhar o mesmo núcleo e ainda serem muito diferentes para usarem o mesmo tripleto para descrevê-los. Por exemplo, o Android executando em um telefone móvel é completamente diferente do Ubuntu executando em um servidor ARM64, apesar de ambos estarem executando no mesmo tipo de CPU (ARM64) e usando o mesmo núcleo (Linux).

Sem uma camada de emulação, você não consegue executar um executável para um servidor em um telefone móvel ou vice versa. Assim, o campo “system” foi dividido nos campos *kernel* e *os* para designar esses sistemas inequivocamente. No nosso exemplo, O sistema Android é designado como `aarch64-unknown-linux-android` e o sistema Ubuntu é designado como `aarch64-unknown-linux-gnu`.

A palavra “tripleto” permanece embutida no léxico. Uma maneira simples para se determinar o teu tripleto do sistema é a de executar o conjunto de comandos sequenciais **config.guess** que vem com o fonte para muitos pacotes. Desempacote os fontes do `binutils`, execute o conjunto de comandos sequenciais `./config.guess` e observe a saída gerada. Por exemplo, para um processador Intel de 32 bits, a saída gerada será `i686-pc-linux-gnu`. Em um sistema de 64 bits, será `x86_64-pc-linux-gnu`. Na maioria dos sistemas Linux, o comando ainda mais simples **gcc -dumpmachine** te dará informação semelhante.

Você também deveria estar ciente do nome do vinculador dinâmico da plataforma, frequentemente referido como o carregador dinâmico (não seja confundido com o vinculador padrão **ld** que é parte do `binutils`). O vinculador dinâmico fornecido pelo pacote `glibc` encontra e carrega as bibliotecas compartilhadas necessárias para um programa, prepara o programa para execução e então o executa. O nome do vinculador dinâmico para uma máquina Intel de 32 bits é `ld-linux.so.2`; e é `ld-linux-x86-64.so.2` em sistemas de 64 bits. Uma maneira infalível para se determinar o nome do vinculador dinâmico é a de inspecionar uma biblioteca aleatória oriunda do sistema anfitrião executando: **readelf -l <nome do binário> | grep interpreter** e observar a saída gerada. A referência autoritativa cobrindo todas as plataformas está em *uma página wiki da Glibc*.

Existem dois pontos-chave para uma compilação cruzada:

- Ao produzir e processar o código de máquina supostamente para ser executado na “anfitriã”, o conjunto cruzado de ferramentas precisa ser usado. Observe que o conjunto nativo de ferramentas originário da “construtora” ainda pode ser invocado para gerar código de máquina supostamente para ser executado na “construtora”. Por exemplo, o sistema de construção possivelmente compile um gerador com o conjunto nativo de ferramentas, então gere um arquivo fonte C com o gerador, e finalmente compile o arquivo fonte C com o conjunto cruzado de ferramentas, de modo que o código gerado estará apto para executar na “anfitriã”.

Com um sistema de construção baseado em autoconf, esse requisito é garantido usando-se a chave `--host` para se especificar o tripleto da “anfitriã”. Com essa chave, o sistema de construção usará os componentes do conjunto de ferramentas prefixados com `<the host triplet>` para gerar e processar o código de máquina para “a anfitriã”; por exemplo, o compilador será `<the host triplet>-gcc` e a ferramenta **readelf** será `<the host triplet>-readelf`.

- O sistema de construção não deveria tentar executar nenhum código de máquina gerado supostamente para ser executado na “anfitriã.” Por exemplo, ao construir um utilitário nativamente, a página de manual dele pode ser gerada executando-se o utilitário com a chave `--help` e processando-se a saída gerada, mas geralmente não é possível fazer isso para uma compilação cruzada, pois o utilitário possivelmente falhe para executar na “construtora”: obviamente é impossível executar código de máquina ARM64 em uma CPU x86 (sem um emulador).

Com um sistema de construção baseado em autoconf, esse requisito é satisfeito no “modo de compilação cruzada”, onde os recursos opcionais que exigem executar código de máquina para “a anfitriã” durante o tempo de construção são desabilitados. Quando o tripleto da “anfitriã” for especificado explicitamente, o “modo de compilação cruzada” é habilitado se e somente se o conjunto de comandos sequenciais **configure** falhar para executar um programa fictício compilado no código de máquina da “anfitriã” ou o tripleto da “construtora” for especificado explicitamente por meio da chave `--build` e ele for diferente do tripleto da “anfitriã”.

Para a finalidade de compilar cruzadamente um pacote para o sistema temporário LFS, o nome do tripleto do sistema é ligeiramente ajustado mudando-se o campo “vendor” na variável `LFS_TGT`, de forma que ele diga “lfs” e `LFS_TGT` é então especificada como o tripleto “da anfitriã” via `--host`, de forma que o conjunto cruzado de ferramentas será usado para gerar e processar o código de máquina executando como parte do sistema temporário LFS. E também nós precisamos habilitar “o modo de compilação cruzada”: apesar do código de máquina da “anfitriã”, ou seja, o código de máquina para o sistema temporário LFS, estar apto para executar na CPU real, ele possivelmente se refira a uma biblioteca não disponível na “construtora” (a distribuição da anfitriã), ou algum código ou dados não existir ou ser definido diferentemente na biblioteca, mesmo se acontecer de estar disponível. Ao compilar cruzadamente um pacote para o sistema temporário LFS, nós não podemos confiar no conjunto de comandos sequenciais **configure** para detectar esse problema com o programa fictício: o fictício usa somente alguns componentes na `libc` que a `libc` da distribuição da anfitriã provavelmente fornece (a menos que, talvez, a distribuição da anfitriã use uma implementação da `libc` diferente, como Musl), de forma que ele não falhará como os programas realmente úteis provavelmente falhariam. Portanto, nós precisamos especificar explicitamente o tripleto da “construtora” para habilitar “o modo de compilação cruzada.” O valor que nós usamos é exatamente o padrão, ou seja, o tripleto original do sistema proveniente da saída gerada do **config.guess**, mas “o modo de compilação cruzada” depende de uma especificação explícita, como nós discutimos.

Nós usamos a opção `--with-sysroot` ao construir o vinculador cruzado e o compilador cruzado, para dizer a eles onde encontrar os arquivos necessários para “a anfitriã.” Isso quase garante que nenhum dos outros programas construídos no Capítulo 6 consiga vincular-se a bibliotecas na “construtora.” A palavra “quase” é usada porque **libtool**, um envolucrador de “compatibilidade” do compilador e do vinculador para sistemas de construção baseados em autoconf, pode tentar ser muito inteligente e passar erroneamente opções permitindo que o vinculador encontre bibliotecas da “construtora.” Para evitar essa precipitação, nós precisamos deletar os arquivos de arquivamento da `libtool` (.la) e consertar uma cópia desatualizada da `libtool` enviada com o código do Binutils.

Estágio	Construtora	Anfitriã	Alvo	Ação
1	pc	pc	lfs	Construir compilador cruzado cc1 usando cc-pc em pc.
2	pc	lfs	lfs	Construir compilador cc-lfs usando cc1 em pc.
3	lfs	lfs	lfs	Reconstrua (e talvez teste) cc-lfs usando cc-lfs no lfs.

Na tabela precedente, “em pc” significa que os comandos são executados em uma máquina usando a distribuição já instalada. “Em lfs” significa que os comandos são executados em um ambiente chroot.

Esse não é ainda o fim da estória. A linguagem C não é meramente um compilador; ela também define uma biblioteca padrão. Neste livro, a biblioteca GNU C, chamada de glibc, é usada (existe uma alternativa, "musl"). Essa biblioteca precisa ser compilada para a máquina LFS; isto é, usando o compilador cruzado cc1. Porém, o próprio compilador usa uma biblioteca interna fornecendo sub rotinas complexas para funções não disponíveis no conjunto de instruções do montador. Essa biblioteca interna é chamada de libgcc e ela precisa ser linkada à biblioteca glibc para ser completamente funcional. Além disso, a biblioteca padrão para C++ (libstdc++) também precisa estar linkada com a glibc. A solução para esse problema de ovo e galinha é a de primeiro se construir uma libgcc degradada baseada em cc1, carecendo de algumas funcionalidades, tais como camadas e manuseio de exceções, e então se construir a glibc usando esse compilador degradado (a glibc em si não é degradada), e também se construir a libstdc++. Essa última biblioteca carecerá de algumas das funcionalidades da libgcc.

O resultado do parágrafo precedente é o de que cc1 está inapto para construir uma libstdc++ totalmente funcional com a libgcc degradada, mas cc1 é o único compilador disponível para se construir as bibliotecas C/C++ durante o estágio 2. Conforme nós discutimos, não podemos executar cc-lfs na pc (a distribuição da anfitriã) porque ele possivelmente exija alguma biblioteca, código ou dados não disponível na “construtora” (a distribuição da anfitriã). Então, quando nós construímos o estágio 2 do gcc, substituímos o caminho de pesquisa da biblioteca para linkar libstdc++ à libgcc recém-reconstruída em vez da construção antiga e degradada. Isso torna a libstdc++ reconstruída totalmente funcional.

No Capítulo 8 (ou “estágio 3”), todos os pacotes necessários para o sistema LFS são construídos. Mesmo se um pacote já tiver sido instalado no sistema LFS em um capítulo anterior, nós ainda reconstruímos o pacote. O principal motivo para se reconstruir esses pacotes é para torná-los estáveis: se reinstalarmos um pacote do LFS em um sistema LFS concluído, o conteúdo reinstalado do pacote deveria ser o mesmo que o conteúdo do mesmo pacote quando instalado pela primeira vez no Capítulo 8. Os pacotes temporários instalados no Capítulo 6 ou no Capítulo 7 não conseguem satisfazer essa exigência, porque alguns recursos opcionais deles são desabilitados devido ou às dependências ausentes ou ao “modo de compilação cruzada.” Além disso, um motivo secundário para se reconstruir os pacotes é para executar as suítes de teste.

Outros Detalhes Procedurais

O compilador cruzado será instalado em um diretório `$LFS/tools` separado, pois ele não será parte do sistema final.

Binutils é instalado primeiro, pois as execuções do **configure** do gcc e da glibc realizam vários testes de recursos no montador e no lincador para determinar quais recursos de software habilitar ou desabilitar. Isso é mais importante do que, inicialmente, alguém possa perceber. Um gcc ou uma glibc configurado incorretamente pode resultar em um conjunto de ferramentas sutilmente quebrado, onde o impacto de tal quebra talvez não se manifeste até próximo do final da construção de uma distribuição inteira. Uma falha de suíte de teste normalmente destacará tal erro antes que muito mais trabalho adicional seja realizado.

Binutils instala o montador e o lincador dele em dois locais, `$LFS/tools/bin` e `$LFS/tools/$LFS_TGT/bin`. As ferramentas em um local são rigidamente lincadas às outras. Uma faceta importante do lincador é a ordem dele de procura de biblioteca. Informação detalhada pode ser obtida a partir do **ld** passando-lhe o sinalizador `--verbose`. Por exemplo, `$LFS_TGT-ld --verbose | grep SEARCH` ilustrará os caminhos atuais de procura e a ordem deles. (Observe que esse exemplo consegue ser executado conforme mostrado somente enquanto logado(a) como usuário `lfs`. Se você retornar para esta página posteriormente, substitua `$LFS_TGT-ld` por `ld`).

O próximo pacote instalado é gcc. Um exemplo do que pode ser visto durante a execução dele do **configure** é:

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

Isso é importante pelas razões mencionadas acima. Também demonstra que o conjunto de comandos sequenciais de configuração do gcc não procura nos diretórios do PATH para encontrar quais ferramentas usar. Entretanto, durante a operação real do gcc em si, os mesmos caminhos de procura não são necessariamente usados. Para descobrir qual lincador padrão o gcc usará, execute: `$LFS_TGT-gcc -print-prog-name=ld`. (Novamente, remova o prefixo `$LFS_TGT-` se retornar a isso posteriormente.)

Informação detalhada pode ser obtida a partir do gcc passando-se a opção de linha de comando `-v` enquanto compilar um programa. Por exemplo, `$LFS_TGT-gcc -v example.c` (ou sem `$LFS_TGT-` se retornar posteriormente) exibirá informação detalhada acerca do pré-processador, compilação e estágios da montagem, incluindo os caminhos de procura do gcc para cabeçalhos inclusos e a ordem deles.

Em seguida: cabeçalhos sanitizados da API do Linux. Eles permitem que a biblioteca C padrão (glibc) interaja com os recursos que o núcleo Linux fornecerá.

Em seguida vem glibc. Esse é o primeiro pacote que nós compilamos cruzadamente. Nós usamos a opção `--host=$LFS_TGT` para fazer o sistema de construção usar aquelas ferramentas prefixadas com `$LFS_TGT-`, e a opção `--build=$(../scripts/config.guess)` para habilitar “o modo de compilação cruzada” como discutimos. A variável `DESTDIR` é usada para forçar a instalação no sistema de arquivos do LFS.

Conforme mencionado acima, a biblioteca padrão C++ é compilada em seguida, seguida no Capítulo 6 por outros programas que precisam ser compilados cruzadamente para quebrar dependências circulares ao tempo da construção. As etapas para aqueles pacotes são semelhantes às etapas para glibc.

No final do Capítulo 6 o compilador nativo do LFS é instalado. Primeiro `binutils-pass2` é construído, no mesmo diretório `DESTDIR` que os outros programas, então a segunda passagem de gcc é construída, omitindo-se algumas bibliotecas não críticas.

Uma vez dentro do ambiente `chroot` no Capítulo 7, as instalações temporárias de programas necessários para a operação apropriada do conjunto de ferramentas são realizadas. Deste ponto em diante, o conjunto central de ferramentas está auto-contido e auto-hospedado. No Capítulo 8, as versões finais de todos os pacotes necessários para um sistema completamente funcional são construídas, testadas e instaladas.

Instruções Gerais de Compilação



Cuidado

Durante um ciclo de desenvolvimento do LFS, as instruções no livro frequentemente são modificadas para se adaptarem a uma atualização de pacote ou para tirar vantagem de novos recursos provenientes de pacotes atualizados. Misturar-se as instruções de diferentes versões do livro LFS pode causar quebras sutis. Esse tipo de problema geralmente é um resultado do reuso de algum conjunto de comandos sequenciais criado para um lançamento anterior do LFS. Tal reuso é fortemente desencorajado. Se você estiver reusando conjuntos de comandos sequenciais para um lançamento anterior do LFS por qualquer motivo, você precisará ter muito cuidado para atualizar os conjuntos de comandos sequenciais para corresponderem à versão atual do livro do LFS.

Aqui estão algumas coisas que você deveria saber a respeito de construir cada pacote:

- Vários pacotes são remendados antes da compilação, porém somente quando o remendo for necessário para contornar um problema. Um remendo frequentemente é necessário tanto neste quanto nos capítulos seguintes, porém às vezes, quando o mesmo pacote for construído mais que uma vez, o remendo não é necessário imediatamente. Portanto, não se preocupe se as instruções para um remendo baixado pareçam estar ausentes. Mensagens de aviso acerca de *offset* ou *fuzz* também possivelmente sejam encontradas ao se aplicar um remendo. Não se preocupe com esses alertas; o remendo ainda foi aplicado com sucesso.
- Durante a compilação da maioria dos pacotes, alguns avisos rolarão na tela. Esses são normais e seguramente podem ser ignorados. Esses alertas usualmente são a respeito do uso de sintaxe C ou C++ obsoleta, porém não inválida. Padrões C mudam com ampla frequência e alguns pacotes ainda não foram atualizados. Esse não é um problema sério, porém causa o aparecimento dos avisos.
- Verifique uma última vez se a variável de ambiente LFS está configurada adequadamente:

```
echo $LFS
```

Certifique-se de que a saída gerada mostra o caminho para o ponto de montagem da partição do LFS, que é `/mnt/lfs`, usando nosso exemplo.

- Finalmente, dois itens importantes precisam ser enfatizados:



Importante

As instruções de construção assumem que as Exigências do Sistema Anfitrião, incluindo links simbólicos, tenham sido configuradas adequadamente:

- **bash** é o shell em uso.
- **sh** é um link simbólico para **bash**.
- **/usr/bin/awk** é um link simbólico para **gawk**.
- **/usr/bin/yacc** é um link simbólico para **bison** ou um pequeno conjunto de comandos sequenciais que executa bison.



Importante

Aqui está uma sinopse do processo de construção.

1. Coloque todos os pacotes e os remendos em um diretório que estará acessível a partir do ambiente chroot, tal como `/mnt/lfs/sources/`.
2. Mude para o diretório `/mnt/lfs/sources/`.
3. Para cada pacote:
 - a. Usando o programa **tar**, extraia o pacote para ser construído. No Capítulo 5 e no Capítulo 6, garanta que você seja o(a) usante *lfs* quando extrair o pacote.

Não use nenhum método, exceto o comando **tar**, para extrair o código fonte. Notadamente, usar o comando **cp -R** para copiar a árvore do código fonte para outro lugar pode destruir carimbos de tempo na árvore do fonte e causar falha de construção.

- b. Mude para o diretório criado quando o pacote foi extraído.
- c. Siga as instruções para construir o pacote.
- d. Mude de volta para o diretório dos fontes quando a construção estiver completa.
- e. Delete o diretório do fonte extraído, a menos que instruído(a) do contrário.

Capítulo 5. Compilando um Conjunto Cruzado de Ferramentas

5.1. Introdução

Este capítulo mostra como construir um compilador cruzado e as ferramentas associadas dele. Embora aqui a compilação cruzada ser falseada, os princípios são os mesmos que para um conjunto cruzado real de ferramentas.

Os programas compilados neste capítulo serão instalados sob o diretório `$LFS/tools` para mantê-los separados dos arquivos instalados nos capítulos seguintes. As bibliotecas, por outro lado, são instaladas no lugar final delas, pois elas pertencem ao sistema que nós queremos construir.

5.2. Binutils-2.47 - Passagem 1

O pacote Binutils contém um vinculador, um montador e outras ferramentas para manusear arquivos objeto.

Tempo aproximado de 1 UPC

construção:

Espaço em disco exigido: 691 MB

5.2.1. Instalação do Binutils Cruzado



Nota

Volte e releia as observações na seção intitulada Instruções Gerais de Compilação. Entender as observações rotuladas como importantes pode salvar você de um monte de problemas depois.

É importante que o Binutils seja o primeiro pacote compilado, pois ambos a Glibc e o GCC realizam vários testes sobre o vinculador e o montador disponíveis para determinar quais dos próprios recursos deles habilitar.

A documentação do Binutils recomenda construir o Binutils em um diretório dedicado à construção:

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



Nota

Para a finalidade de que os valores da UPC listados no resto do livro sejam de algum uso, meça o tempo que leva para construir esse pacote desde a configuração até, e incluindo, a primeira instalação. Para fazer isso facilmente, encapsule os comandos em um comando **time** desta forma: **time { ../configure ... && make && make install; }**.

Agora prepare o Binutils para compilação:

```
../configure --prefix=$LFS/tools \
             --with-sysroot=$LFS \
             --target=$LFS_TGT \
             --disable-nls \
             --enable-gprofng=no \
             --disable-werror \
             --enable-new-dtags \
             --enable-default-hash-style=gnu
```

O significado das opções do configure:



Nota

Ao contrário de outros pacotes, nem todas as opções listadas abaixo aparecem ao executar **./configure --help**. Por exemplo, para encontrar a opção **--with-sysroot**, você tem que executar **ld/configure --help**. Todas as opções podem ser listadas de uma só vez com **./configure --help=recursive**.

--prefix=\$LFS/tools

Isso diz para o script configure para preparar para instalar os aplicativos do Binutils no diretório **\$LFS/tools**.

--with-sysroot=\$LFS

Para compilação cruzada, isso diz ao sistema de construção para procurar em **\$LFS** pelas bibliotecas alvo de sistema conforme necessário.

--target=\$LFS_TGT

Por causa da descrição de máquina na variável **LFS_TGT** ser ligeiramente diferente do valor retornado pelo script **config.guess**, essa chave dirá ao script **configure** para ajustar o sistema de construção do binutils para construir um vinculador cruzado.

`--disable-nls`

Isso desabilita internacionalização, uma vez que i18n não é necessária para as ferramentas temporárias.

`--enable-gprofng=no`

Isso desabilita a construção do gprofng o qual não é necessário para as ferramentas temporárias.

`--disable-werror`

Isso evita que a construção pare no caso de existirem alertas originários do compilador do anfitrião.

`--enable-new-dtags`

Isso faz com que o vinculador use a etiqueta “runpath” para incorporar caminhos de pesquisa de biblioteca em executáveis e em bibliotecas compartilhadas, em vez da tradicional etiqueta “rpath”. Ela torna mais fácil depurar executáveis vinculados dinamicamente e contorna possíveis problemas na suíte de teste de alguns pacotes.

`--enable-default-hash-style=gnu`

Por padrão, o vinculador geraria a tabela "hash" no estilo "GNU" e a tabela "hash" "ELF" clássica para bibliotecas compartilhadas e executáveis vinculados dinamicamente. As tabelas "hash" destinam-se somente a um vinculador dinâmico para realizar a pesquisa de símbolos. No LFS, o vinculador dinâmico (fornecido pelo pacote "Glibc") sempre usará a tabela "hash" no estilo "GNU", que é mais rápida de consultar. Portanto, a tabela "hash" "ELF" clássica é completamente inútil. Isso faz com que o vinculador gere somente a tabela "hash" estilo "GNU" por padrão, de forma que possamos evitar desperdiçar tempo para gerar a tabela "hash" "ELF" clássica quando construirmos os pacotes ou desperdiçar espaço em disco para armazená-la.

Continue compilando o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 8.22.2, “Conteúdo do Binutils.”

5.3. GCC-16.2.0 - Passagem 1

O pacote GCC contém a GNU Compiler Collection, a qual inclui os compiladores C e C++.

Tempo aproximado de construção: 4,3 UPC

Espaço em disco exigido: 5,8 GB

5.3.1. Instalação do GCC Cruzado

O GCC exige os pacotes GMP, MPFR e MPC. Uma vez que esses pacotes talvez não estejam incluídos na sua distribuição anfitriã, eles serão construídos com o GCC. Desempacote cada pacote dentro do diretório de fonte do GCC e renomeie os diretórios resultantes, de forma que os procedimentos de construção do GCC automaticamente os usarão:



Nota

Existem mal-entendidos frequentes a respeito das instruções aqui. Os procedimentos são os mesmos que os de cada um dos outros pacotes, conforme explicados anteriormente (Instruções de construção de pacote). Primeiro, extraia o tarball gcc-16.2.0 a partir do diretório dos fontes e então mude para o diretório criado. Somente então deveria você prosseguir com as instruções abaixo.

```
tar -xf ../mpfr-4.2.2.tar.xz
mv -v mpfr-4.2.2 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.4.1.tar.xz
mv -v mpc-1.4.1 mpc
```

Em anfitriões x86_64, configure o nome padrão de diretório para bibliotecas de 64 bits para “lib”:

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



Nota

Esse exemplo demonstra o uso da chave *-i.orig*. Ela faz com que o **sed** copie o arquivo `t-linux64` para `t-linux64.orig` e, então, edite o arquivo `t-linux64` original no local. Assim, você pode executar **diff -u gcc/config/i386/t-linux64{.orig,}** para visualizar a mudança feita pelo comando **sed** posteriormente. Nós usaremos simplesmente *-i* (que somente edita o arquivo original no local, sem copiá-lo) para todos os outros pacotes do livro, mas você pode mudá-lo para *-i.orig* em qualquer caso que queira manter uma cópia do arquivo original.

A documentação do GCC recomenda construir o GCC em um diretório de construção dedicado:

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

Prepare o GCC para compilação:

```
../configure \
--target=$LFS_TGT \
--prefix=$LFS/tools \
--with-glibc-version=2.44 \
--with-sysroot=$LFS \
--with-newlib \
--without-headers \
--enable-default-pie \
--enable-default-ssp \
--disable-fixincludes \
--disable-nls \
--disable-shared \
--disable-multilib \
--disable-threads \
--disable-libatomic \
--disable-libgomp \
--disable-libquadmath \
--disable-libssp \
--disable-libvtv \
--disable-libstdcxx \
--enable-languages=c,c++
```

O significado das opções do configure:

--with-glibc-version=2.44

Essa opção especifica a versão da Glibc que será usada no alvo. Ela não é relevante para a libc da distribuição anfitriã, pois tudo compilado pela passagem 1 do GCC executará no ambiente chroot, o qual é isolado da libc da distribuição anfitriã.

--with-newlib

Uma vez que uma biblioteca C funcional ainda não está disponível, isso assegura que a constante `inhibit_libc` esteja definida quando da construção de `libgcc`. Isso evita a compilação de qualquer código que exija suporte à libc.

--without-headers

Quando da criação de um compilador cruzado completo, o GCC exige cabeçalhos padrão compatíveis com o sistema alvo. Para nossos propósitos esses cabeçalhos não serão necessários. Essa chave evita que o GCC procure por eles.

--enable-default-pie e --enable-default-ssp

Essas chaves permitem que o GCC compile aplicativos com alguns recursos de segurança reforçados (mais informação a respeito delas na observação a respeito de PIE e SSP no capítulo 8) por padrão. Elas não são estritamente necessárias neste estágio, pois o compilador produzirá apenas executáveis temporários. Mas, é mais limpo ter os pacotes temporários o mais próximo possível dos finais.

--disable-fixincludes

Por padrão, durante a instalação do GCC, alguns cabeçalhos do sistema seriam “corrigidos” para serem usados com o GCC. Neste ponto, nós não instalamos nenhum cabeçalho para o compilador cruzado usar, de forma que construir os programas auxiliares para a “correção” apenas desperdiçaria tempo. Esta chave desabilita a “correção” e ignora a construção desses programas.

--disable-shared

Essa chave força o GCC a vincular as bibliotecas internas dele estaticamente. Nós precisamos disso, pois as bibliotecas compartilhadas exigem a Glibc, que ainda não está instalada no sistema alvo.

--disable-multilib

Em `x86_64`, o LFS não suporta uma configuração multi bibliotecas. Essa chave é inofensiva para `x86`.

```
--disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp,
--disable-libvtv e --disable-libstdcxx
```

Essas chaves desabilitam o suporte para threading, libatomic, libgomp, libquadmath, libssp, libvtv e à biblioteca padrão C++ respectivamente. Esses recursos possivelmente falhem para compilar quando da construção de um compilador cruzado e não são necessários para a tarefa de compilar cruzadamente a libc temporária.

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

Essa opção garante que apenas os compiladores C e C++ sejam construídos. Essas são as únicas linguagens necessárias agora.

Compile o GCC executando:

```
make
```

Instale o pacote:

```
make install
```

Essa construção do GCC instalou um par de cabeçalhos internos de sistema. Normalmente um deles, `limits.h`, sequencialmente incluiria o correspondente cabeçalho de sistema `limits.h`, nesse caso, `$LFS/usr/include/limits.h`. Entretanto, ao tempo dessa construção do GCC, `$LFS/usr/include/limits.h` não existe, de forma que o cabeçalho interno que tenha sido instalado é um arquivo parcial, auto-contido, e não inclui os recursos estendidos do cabeçalho de sistema. Isso é adequado para a construção da Glibc, porém o cabeçalho interno completo será necessário posteriormente. Crie uma versão completa do cabeçalho interno usando um comando que é idêntico ao que o sistema de construção do GCC faz em circunstâncias normais:

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

Detalhes acerca deste pacote estão localizados na Seção 8.32.2, “Conteúdo do GCC.”

5.4. Cabeçalhos da API do Linux-7.1.8

Os Cabeçalhos da API do Linux (em linux-7.1.8.tar.xz) expõem a API do kernel para uso pela Glibc.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 1,8 GB

5.4.1. Instalação dos Cabeçalhos da API do Linux

O kernel Linux precisa expor uma Interface de Programação de Aplicativos (API) para a biblioteca C do sistema (Glibc no LFS) usar. Isso é feito por meio de sanitizar vários arquivos de cabeçalho C que são embarcados no tarball do fonte do kernel Linux.

Certifique-se de que não existem arquivos obsoletos embutidos no pacote:

```
make mrproper
```

Agora extraia os cabeçalhos de núcleo visíveis para usante a partir do fonte. O alvo recomendado do make “headers_install” não pode ser usado, pois ele exige rsync, que possivelmente não esteja disponível. Os cabeçalhos são primeiro colocados em `./usr`, então copiados para o local necessário.

```
make headers
find usr/include -type f ! -name '*.h' -delete
cp -rv usr/include $LFS/usr
```

5.4.2. Conteúdo dos Cabeçalhos da API do Linux

Cabeçalhos instalados: `/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` e `/usr/include/xen/*.h`

Diretórios instalados: `/usr/include/asm`, `/usr/include/asm-generic`, `/usr/include/drm`, `/usr/include/linux`, `/usr/include/misc`, `/usr/include/mtd`, `/usr/include/rdma`, `/usr/include/scsi`, `/usr/include/sound`, `/usr/include/video` e `/usr/include/xen`

Descrições Curtas

<code>/usr/include/asm/*.h</code>	Os cabeçalhos ASM da API do Linux
<code>/usr/include/asm-generic/*.h</code>	Os cabeçalhos genéricos ASM da API do Linux
<code>/usr/include/drm/*.h</code>	Os cabeçalhos DRM da API do Linux
<code>/usr/include/linux/*.h</code>	Os cabeçalhos Linux da API do Linux
<code>/usr/include/misc/*.h</code>	Os cabeçalhos diversos da API do Linux
<code>/usr/include/mtd/*.h</code>	Os cabeçalhos MTD da API do Linux
<code>/usr/include/rdma/*.h</code>	Os cabeçalhos RDMA da API do Linux
<code>/usr/include/scsi/*.h</code>	Os cabeçalhos SCSI da API do Linux
<code>/usr/include/sound/*.h</code>	Os cabeçalhos de som da API do Linux
<code>/usr/include/video/*.h</code>	Os cabeçalhos de vídeo da API do Linux
<code>/usr/include/xen/*.h</code>	Os cabeçalhos Xen da API do Linux

5.5. Glibc-2.44

O pacote Glibc contém a principal biblioteca C. Essa biblioteca fornece as rotinas básicas para alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos, manuseio de sequências de caracteres, correspondência de padrões, aritmética, e daí por diante.

Tempo aproximado de construção: 1,3 UPC

Espaço em disco exigido: 900 MB

5.5.1. Instalação da Glibc

Primeiro, crie um link simbólico para conformidade com a LSB. Adicionalmente, para x86_64, crie um link simbólico de compatibilidade exigido para a operação adequada do carregador dinâmico de biblioteca:

```
case $(uname -m) in
    i?86) ln -sfv ld-linux.so.2 $LFS/lib/ld-lsb.so.3
    ;;
    x86_64) ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64
            ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64/ld-lsb-x86-64.so.3
    ;;
esac
```



Nota

O comando acima está correto. O comando **"ln"** tem várias versões sintáticas, de forma que tenha certeza de verificar **"info coreutils ln"** e **"ln(1)"** antes de informar o que possivelmente aparente ser um erro.

Alguns dos aplicativos Glibc usam o diretório não conforme com a FHS `/var/db` para armazenar os dados em tempo de execução deles. Aplique o seguinte remendo para fazer com que tais aplicativos armazenem os dados em tempo de execução deles nos locais conformes com a FHS:

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

Corrija um problema que faz com que a função *tanh(3)* quebre em alguns processadores x86_64 antigos e corrija instalar com múltiplas tarefas do **make**:

```
patch -Np1 -i ../glibc-2.44-upstream_fixes-1.patch
```

A documentação da Glibc recomenda construir a Glibc em um diretório dedicado à construção:

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

Assegure que os utilitários **ldconfig** e **sln** sejam instalados em `/usr/sbin`:

```
echo "rootsbindir=/usr/sbin" > configparms
```

A seguir, prepare a Glibc para compilação:

```
../configure \
  --prefix=/usr \
  --host=$LFS_TGT \
  --build=$(../scripts/config.guess) \
  --disable-nscd \
  libc_cv_slibdir=/usr/lib \
  --enable-kernel=5.10
```

O significado das opções do configure:

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

O efeito combinado dessas chaves é o de que o sistema de construção da Glibc se autoconfigura para ser compilado cruzadamente, usando o vinculador cruzado e o compilador cruzado em `$LFS/tools`.

```
--enable-kernel=5.10
```

Isso diz para a Glibc para compilar a biblioteca com suporte para núcleos Linux 5.10 e posteriores. Contornos para núcleos mais antigos não estão habilitados.

```
libc_cv_slibdir=/usr/lib
```

Isso garante que a biblioteca seja instalada em `/usr/lib` em vez do padrão `/lib64` em máquinas de 64 bits.

```
--disable-nscd
```

Não construa o processo de segundo plano de armazenamento temporário do serviço de nomes, o qual não mais é usado.

Durante este estágio o seguinte aviso pode aparecer:

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

O ausente ou incompatível aplicativo **msgfmt** geralmente é inofensivo. Esse aplicativo **msgfmt** é parte do pacote Gettext, que a distribuição anfitriã deveria fornecer.



Nota

Tem havido informes de que esse pacote possivelmente falhe ao construir-se como um “make paralelo”. Se isso ocorrer, reexecute o comando make com a opção `-j1`.

Compile o pacote:

```
make
```

Instale o pacote:



Atenção

Se `LFS` não estiver adequadamente configurada, e a despeito das recomendações, você estiver construindo como `root`, [então] o próximo comando instalará a recém construída Glibc em seu sistema anfitrião, o que quase certamente o tornará inutilizável. Portanto, verifique duas vezes se o ambiente está corretamente configurado e que você não é o(a) `root` antes de executar o seguinte comando.

```
make DESTDIR=$LFS install
```

O significado da opção `make install`:

```
DESTDIR=$LFS
```

A variável `DESTDIR` do `make` é usada por quase todos os pacotes para definir o local onde o pacote deveria ser instalado. Se ela não estiver configurada, padroniza para o diretório raiz (`/`). Aqui nós especificamos que o pacote seja instalado em `$LFS`, que se tornará o diretório raiz na Seção 7.4, “Entrando no Ambiente Chroot.”

Corrija caminho codificado rigidamente para o carregador de executável no script **ldd**:

```
sed '/RTLDLIST=/s@/usr@@g' -i $LFS/usr/bin/ldd
```

Agora que nosso conjunto cruzado de ferramentas está no lugar, é importante garantir que compilação e vinculação funcionarão conforme esperado. Nós fazemos isso realizando algumas verificações de sanidade:

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

Não deveriam existir erros, e a saída gerada do último comando será (levando em consideração diferenças específicas da plataforma no nome do vinculador dinâmico):

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

Observe que esse caminho não deveria conter `/mnt/lfs` (ou o valor da variável `LFS` se você usou uma diferente). O caminho é resolvido quando o programa compilado for executado, e isso só deveria acontecer depois que nós entrarmos no ambiente `chroot` onde o núcleo consideraria `$LFS` como o diretório raiz (`/`).

Agora certifique-se de que nós estamos configurados para usar os arquivos corretos de iniciação:

```
grep -E -o "$LFS/lib.*S?crt[1in].*succeeded" dummy.log
```

A saída gerada do último comando deveria ser:

```
/mnt/lfs/lib/../../lib/Scrt1.o succeeded
/mnt/lfs/lib/../../lib/crti.o succeeded
/mnt/lfs/lib/../../lib/crtn.o succeeded
```

Verifique se o compilador está procurando os arquivos corretos de cabeçalho:

```
grep -B3 "^ $LFS/usr/include" dummy.log
```

Esse comando deveria retornar a seguinte saída gerada:

```
#include <...> search starts here:
/mnt/lfs/tools/lib/gcc/x86_64-lfs-linux-gnu/16.2.0/include
/mnt/lfs/tools/lib/gcc/x86_64-lfs-linux-gnu/16.2.0/include-fixed
/mnt/lfs/usr/include
```

Novamente, o diretório nomeado depois do teu tripleto alvo possivelmente seja diferente do acima, dependendo da arquitetura do teu sistema.

Agora, verifique se o novo vinculador está sendo usado com os caminhos corretos de procura:

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

Referências para caminhos que tenham componentes com `-linux-gnu` deveriam ser ignoradas, porém, do contrário, a saída gerada do último comando deveria ser:

```
SEARCH_DIR(="/mnt/lfs/tools/x86_64-lfs-linux-gnu/lib64")
SEARCH_DIR(="/usr/local/lib64")
SEARCH_DIR(="/lib64")
SEARCH_DIR(="/usr/lib64")
SEARCH_DIR(="/mnt/lfs/tools/x86_64-lfs-linux-gnu/lib")
SEARCH_DIR(="/usr/local/lib")
SEARCH_DIR(="/lib")
SEARCH_DIR(="/usr/lib");
```

Um sistema de 32 bits possivelmente use alguns outros diretórios, mas de qualquer maneira o aspecto importante aqui é que todos os caminhos deveriam começar com um sinal de igual (`=`), que seria substituído pelo diretório `sysroot` que nós configuramos para o vinculador.

Em seguida, tenha certeza de que nós estamos usando a `libc` correta:

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

A saída gerada do último comando deveria ser:

```
attempt to open /mnt/lfs/usr/lib/libc.so.6 succeeded
```

Tenha certeza de que `GCC` está usando o vinculador dinâmico correto:

```
grep found dummy.log
```

A saída gerada do último comando deveria ser (levando em consideração diferenças específicas da plataforma no nome do vinculador dinâmico):

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

Se a saída gerada não aparecer conforme mostrado acima ou não for recebida, então alguma coisa está seriamente errada. Investigue e refaça as etapas para descobrir onde está o problema e corrija-o. Quaisquer problemas deveriam ser resolvidos antes de se continuar com o processo.

Uma vez que tudo esteja funcionando corretamente, remova os arquivos de teste:

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



Nota

Construir os pacotes no próximo capítulo servirá como uma verificação adicional de que o conjunto de ferramentas foi construído adequadamente. Se algum pacote, especialmente o Binutils-passagem 2 ou o GCC-passagem 2, falhar na construção, [então] isso é uma indicação de que alguma coisa deu errado com as instalações anteriores do Binutils, GCC ou da Glibc.

Detalhes acerca desse pacote estão localizados na Seção 8.5.3, “Conteúdo do Glibc.”

5.6. Libstdc++ oriundo de GCC-16.2.0

Libstdc++ é a biblioteca padrão C++. Ela é necessária para compilar código C++ (parte de GCC é escrito em C++), porém nós tivemos que adiar a instalação dela quando construímos gcc-pass1, pois a Libstdc++ depende da Glibc, que ainda não estava disponível no diretório alvo.

Tempo aproximado de 0,3 UPC

construção:

Espaço em disco exigido: 1,5 GB

5.6.1. Instalação da Libstdc++ Alvo



Nota

Libstdc++ é parte dos fontes do GCC. Você deveria primeiro desempacotar o tarball do GCC e mudar para o diretório gcc-16.2.0.

Crie um diretório separado de construção para a Libstdc++ e entre nele:

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

Prepare a Libstdc++ para compilação:

```
../libstdc++-v3/configure \
--host=$LFS_TGT \
--build=$(../config.guess) \
CXX=$LFS_TGT-gcc \
--prefix=/usr \
--disable-multilib \
--disable-nls \
--disable-libstdcxx-pch \
--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/16.2.0
```

O significado das opções do configure:

`--host=...`

Especifica que o compilador cruzado que nós acabamos de construir deveria ser usado em vez daquele em /usr/bin.

`CXX=$LFS_TGT-gcc`

Especifica o uso de `$LFS_TGT-gcc` para compilar o código C++. O padrão, `$LFS_TGT-g++`, especificaria automaticamente a ligação com a libstdc++ ao invocar o linkador; no entanto, no nosso caso, a libstdc++ ainda não está instalada, de modo que algumas verificações no conjunto de comandos sequenciados **configure** apresentariam comportamento inadequado com o `$LFS_TGT-g++`. Em uma construção normal, essa substituição é passada automaticamente para o conjunto de comandos sequenciados **configure** a partir do diretório de nível superior. Aqui, nós a passamos explicitamente.

`--disable-libstdcxx-pch`

Essa chave evita a instalação de arquivos pré-compilados include, os quais não são necessários neste estágio.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/16.2.0`

Isso especifica o diretório de instalação para arquivos include. Por causa da Libstdc++ ser a biblioteca padrão C++ para o LFS, esse diretório deveria corresponder com o local onde o compilador C++ (`$LFS_TGT-g++`) procuraria pelos arquivos padrão include C++. Em uma construção normal, essa informação é automaticamente passada para as opções **configure** da Libstdc++ a partir do diretório de nível de topo. Em nosso caso, essa informação precisa ser explicitamente dada. O compilador C++ precederá o caminho raiz do sistema `$LFS` (especificado quando da construção do GCC passagem 1) para o caminho de pesquisa de arquivo include, de

forma que ele atualmente pesquisará em `$LFS/tools/$LFS_TGT/include/c++/16.2.0`. A combinação da variável `DESTDIR` (no comando **make install** abaixo) e essa chave causa os cabeçalhos serem instalados lá.

Compile a Libstdc++ executando:

```
make
```

Instale a biblioteca:

```
make DESTDIR=$LFS install
```

Remova os arquivos de arquivamento do libtool, pois eles são danosos para compilação cruzada:

```
rm -v $LFS/usr/lib/lib{stdc++{,exp,fs},supc++}.la
```

Detalhes acerca deste pacote estão localizados na Seção 8.32.2, “Conteúdo do GCC.”

Capítulo 6. Compilando Cruzadamente Ferramentas Temporárias

6.1. Introdução

Este capítulo mostra como compilar cruzadamente utilitários básicos usando o recém construído conjunto de ferramentas cruzados. Esses utilitários são instalados no local final deles, porém ainda não podem ser usados. Tarefas básicas ainda dependem das ferramentas do anfitrião. Apesar disso, as bibliotecas instaladas são usadas quando da vinculação.

Usar os utilitários será possível no próximo capítulo após entrada no ambiente “chroot”. Porém, todos os pacotes construídos no presente capítulo precisam ser construídos antes que façamos isso. Dessa forma nós ainda não podemos ficar independentes do sistema anfitrião.

Uma vez mais, permita-nos relembrar que a configuração inapropriada de `LFS` junto com a construção como `root`, possivelmente torne teu computador não usável. Este capítulo inteiro precisa ser feito como usante `lfs`, com o ambiente conforme descrito na Seção 4.4, “Configurando o Ambiente.”

6.2. M4-1.4.21

O pacote M4 contém um processador de macro.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 39 MB

6.2.1. Instalação do M4

Certifique-se de que os pacotes que usam o gnlub detectem algumas funções mais recentes encontradas na glibc-2.44.

```
cat > $LFS/usr/share/config.site << EOF
ac_cv_func_posix_spawn_file_actions_addchdir=yes
ac_cv_func_posix_spawn_file_actions_addfchdir=yes
EOF
```

Prepare o M4 para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.14.2, “Conteúdo de M4.”

6.3. Ncurses-6.6

O pacote Ncurses contém bibliotecas para manuseio independente de terminal das telas de caracteres .

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 54 MB

6.3.1. Instalação do Ncurses

Primeiro, execute os seguintes comandos para construir o programa **tic** no anfitrião de construção. Nós o instalamos em `$LFS/tools` de forma que ele seja encontrado no `PATH` quando necessário:

```
mkdir build
pushd build
  ../configure --prefix=$LFS/tools AWK=gawk
  make -C include
  make -C progs tic
  install progs/tic $LFS/tools/bin
popd
```

Prepare o Ncurses para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(./config.guess) \
            --mandir=/usr/share/man \
            --with-manpage-format=normal \
            --with-shared \
            --without-normal \
            --with-cxx-shared \
            --without-debug \
            --without-ada \
            --disable-stripping \
            AWK=gawk
```

O significado das novas opções de configurar:

`--with-manpage-format=normal`

Isso evita que o Ncurses instale páginas de manual comprimidas, o que possivelmente aconteça se a própria distribuição anfitriã tiver páginas de manual comprimidas.

`--with-shared`

Isso faz com que o Ncurses construa e instale bibliotecas C compartilhadas.

`--without-normal`

Isso impede o Ncurses de construir e de instalar bibliotecas C estáticas.

`--without-debug`

Isso impede o Ncurses de construir e de instalar bibliotecas de depuração.

`--with-cxx-shared`

Isso faz com que o Ncurses construa e instale vínculos C++ compartilhados. Também evita a construção e instalação de vínculos C++ estáticos.

`--without-ada`

Isso assegura que o Ncurses não construa suporte para o compilador Ada, o qual possivelmente esteja presente no anfitrião, porém não estará disponível até que nós entremos no ambiente **chroot**.

`--disable-stripping`

Essa chave impede o sistema de construção de usar o aplicativo **strip** oriundo do anfitrião. Usar ferramentas do anfitrião em aplicativos compilados cruzadamente pode causar falha.

Awk=gawk

Essa chave impede o sistema de construção de usar o programa **mawk** oriundo do anfitrião. Algumas versões do **mawk** podem causar esse pacote falhar para construir.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
ln -sv libncursesw.so $LFS/usr/lib/libncurses.so
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i $LFS/usr/include/curses.h
```

O significado das opções de instalação:

ln -sv libncursesw.so \$LFS/usr/lib/libncurses.so

A biblioteca `libncurses.so` é necessária para uns poucos pacotes que nós construiremos breve. Nós criamos esse link simbólico para usar a `libncursesw.so` como uma substituta.

sed -e 's/^#if.*XOPEN.*\$/#if 1/' ...

O arquivo de cabeçalho `curses.h` contém a definição de várias estruturas de dados do Ncurses. Com diferentes definições de macro de pré processador, dois conjuntos de definição de estrutura de dados podem ser usados: a definição de oito bits é compatível com a `libncurses.so` e a definição de caracteres largos é compatível com a `libncursesw.so`. Como estamos usando a `libncursesw.so` como uma substituta da `libncurses.so`, edite o arquivo de cabeçalho de forma que ele sempre usará a definição de estrutura de dados de caracteres largos compatível com a `libncursesw.so`.

Detalhes acerca deste pacote estão localizados na Seção 8.33.2, “Conteúdo do Ncurses.”

6.4. Bash-5.3

O pacote Bash contém o Bourne-Again Shell.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 73 MB

6.4.1. Instalação do Bash

Prepare o Bash para compilação:

```
./configure --prefix=/usr \
            --build=$(sh support/config.guess) \
            --host=$LFS_TGT \
            --without-bash-malloc \
            --docdir=/usr/share/doc/bash-5.3
```

O significado das opções do configure:

--without-bash-malloc

Essa opção desliga o uso da função de alocação de memória do Bash (`malloc`) a qual é conhecida por causar falhas de segmentação. Ao se desligar essa opção, o Bash usará as funções `malloc` originárias da Glibc que são mais estáveis.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Faça um link para os aplicativos que usam **sh** para um shell:

```
ln -sv bash $LFS/bin/sh
```

Detalhes acerca deste pacote estão localizados na Seção 8.39.2, “Conteúdo do Bash.”

6.5. Coreutils-9.11

O pacote Coreutils contém aplicativos utilitários básicos necessitados por cada sistema operacional.

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 193 MB

6.5.1. Instalação do Coreutils

Prepare o Coreutils para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess) \
            --enable-install-program=hostname
```

O significado das opções do configure:

--enable-install-program=hostname

Isso habilita o binário **hostname** para ser construído e instalado – ele é desabilitado por padrão, porém é exigido pela suíte de teste do Perl.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Mova aplicativos para os locais finais deles esperados. Apesar de isso não ser necessário neste ambiente temporário, nós precisamos fazer isso, pois alguns aplicativos codificam rigidamente locais de executável:

```
mv -v $LFS/usr/bin/chroot $LFS/usr/sbin
mkdir -pv $LFS/usr/share/man/man8
mv -v $LFS/usr/share/man/man1/chroot.1 $LFS/usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/' $LFS/usr/share/man/man8/chroot.8
```

Detalhes acerca deste pacote estão localizados na Seção 8.61.2, “Conteúdo do Coreutils.”

6.6. Diffutils-3.12

O pacote Diffutils contém aplicativos que mostram as diferenças entre arquivos ou diretórios.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 35 MB

6.6.1. Instalação do Diffutils

Prepare o Diffutils para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            gl_cv_func_strcasecmp_works=yes \
            --build=$(./build-aux/config.guess)
```

O significado das opções do configure:

gl_cv_func_strcasecmp_works=yes

Essa opção especifica o resultado de uma verificação para a função `strcasecmp`. A verificação exige executar um programa compilado em C, e isso é impossível durante a compilação cruzada, pois, em geral, um programa compilado cruzadamente não consegue executar na distribuição anfitriã. Normalmente, para tal verificação, o conjunto de comandos sequenciais **configure** usaria um valor residual para compilação cruzada, mas o valor residual para essa verificação está ausente e o conjunto de comandos sequenciais **configure** não teria valor para usar e apresentaria um erro. O fluxo de desenvolvimento já corrigiu o problema, mas para aplicar a correção, nós precisaríamos executar **autoconf**, que a distribuição anfitriã possivelmente careça. Portanto, nós apenas especificamos o resultado da verificação (*yes*, pois nós sabemos que a função `strcasecmp` na Glibc-2.44 funciona bem), então **configure** apenas usará o valor especificado e ignorará a verificação.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.62.2, “Conteúdo do Diffutils.”

6.7. File-5.48

O pacote File contém um utilitário para determinar o tipo de um dado arquivo ou arquivos.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 46 MB

6.7.1. Instalação do File

O comando **file** no anfitrião de construção precisa ser da mesma versão que aquele que nós estamos construindo com a finalidade de criar o arquivo de assinatura. Execute os seguintes comandos para produzir uma cópia temporária do comando **file**:

```
mkdir build
pushd build
  ../configure --disable-bzlib      \
               --disable-libseccomp \
               --disable-xzlib     \
               --disable-zlib
make
popd
```

O significado da nova opção do configure:

*--disable-**

O script de configuração tenta usar alguns pacotes originários da distribuição anfitriã se os arquivos de biblioteca correspondentes existirem. Isso possivelmente cause falha de compilação se um arquivo de biblioteca existir, porém os arquivos de cabeçalhos correspondentes não. Essas opções evitam usar essas capacidades desnecessárias oriundas do anfitrião.

Prepare o File para compilação:

```
./configure --prefix=/usr --host=$LFS_TGT --build=$(./config.guess)
```

Compile o pacote:

```
make FILE_COMPILE=$(pwd)/build/src/file
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Remova o arquivo de arquivamento do libtool pois ele é danoso para compilação cruzada:

```
rm -v $LFS/usr/lib/libmagic.1a
```

Detalhes acerca deste pacote estão localizados na Seção 8.11.2, “Conteúdo do File.”

6.8. Findutils-4.11.0

O pacote Findutils contém aplicativos para encontrar arquivos. Os aplicativos são fornecidos para procurar ao longo de todos os arquivos em uma árvore de diretórios e para criar, manter e buscar uma base de dados (geralmente mais rápido que o find recursivo, porém não é confiável, a menos que a base de dados tenha sido atualizada recentemente). O Findutils também abastece o aplicativo **xargs**, o qual pode ser usado para executar um comando especificado sobre cada arquivo selecionado por uma pesquisa.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 51 MB

6.8.1. Instalação do Findutils

Prepare o Findutils para compilação:

```
./configure --prefix=/usr          \
            --localstatedir=/var/lib/locate \
            --host=$LFS_TGT         \
            --build=$(build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.63.2, “Conteúdo do Findutils.”

6.9. Gawk-5.4.1

O pacote Gawk contém aplicativos para manipular arquivos de texto.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 52 MB

6.9.1. Instalação do Gawk

Primeiro, garanta que alguns arquivos desnecessários não sejam instalados:

```
sed -i 's/extras//' Makefile.in
```

Prepare o Gawk para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.31.2, “Conteúdo do Gawk.”

6.10. Grep-3.12

O pacote Grep contém aplicativos para procura ao longo do conteúdo de arquivos.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 32 MB

6.10.1. Instalação do Grep

Prepare o Grep para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(./build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.38.2, “Conteúdo do Grep.”

6.11. Gzip-1.14

O pacote Gzip contém aplicativos para comprimir e descomprimir arquivos.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 12 MB

6.11.1. Instalação do Gzip

Prepare o Gzip para compilação:

```
./configure --prefix=/usr --host=$LFS_TGT
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.66.2, “Conteúdo do Gzip.”

6.12. Make-4.4.1

O pacote Make contém um aplicativo para controlar a geração de executáveis e outros arquivos não fonte de um pacote a partir de arquivos fonte.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 15 MB

6.12.1. Instalação do Make

Prepare o Make para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.70.2, “Conteúdo do Make.”

6.13. Patch-2.8

O pacote Patch contém um aplicativo para modificar ou criar arquivos por aplicação de um arquivo “remendo” tipicamente criado pelo aplicativo **diff**.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 14 MB

6.13.1. Instalação do Patch

Prepare o Patch para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.71.2, “Conteúdo do Patch.”

6.14. Sed-4.10

O pacote Sed contém um editor de fluxo.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 27 MB

6.14.1. Instalação do Sed

Prepare o Sed para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(./build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.34.2, “Conteúdo do Sed.”

6.15. Tar-1.35

O pacote Tar fornece a habilidade para criar arquivamentos tar bem como para realizar vários outros tipos de manipulação de arquivamento. Tar pode ser usado sobre arquivamentos previamente criados para extrair arquivos, para armazenar arquivos adicionais ou para atualizar ou listar arquivos que já foram armazenados.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 43 MB

6.15.1. Instalação do Tar

Prepare o Tar para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Detalhes acerca deste pacote estão localizados na Seção 8.72.2, “Conteúdo do Tar.”

6.16. Xz-5.8.3

O pacote Xz contém aplicativos para comprimir e descomprimir arquivos. Ele fornece recursos para os formatos de compressão lzma e o mais novo xz. Comprimir arquivos de texto com **xz** gera uma melhor percentagem de compressão que com os tradicionais comandos **gzip** ou **bzip2**.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 24 MB

6.16.1. Instalação do Xz

Prepare o Xz para compilação:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess) \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.8.3
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Remova o arquivo de arquivamento do libtool pois ele é danoso para compilação cruzada:

```
rm -v $LFS/usr/lib/liblzma.la
```

Detalhes acerca deste pacote estão localizados na Seção 8.8.2, “Conteúdo do Xz.”

6.17. Binutils-2.47 - Passagem 2

O pacote Binutils contém um vinculador, um montador e outras ferramentas para manusear arquivos objeto.

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 560 MB

6.17.1. Instalação do Binutils

O sistema de construção do "Binutils" depende de uma cópia enviada da "libtool" para vincular-se a bibliotecas estáticas internas, mas as cópias "libiberty" e "zlib" enviadas no pacote não usam a "libtool". Essa inconsistência possivelmente cause binários produzidos vinculados erroneamente a bibliotecas originárias da distribuição anfitriã. Contorne esse problema:

```
sed '6031s/$add_dir//' -i ltmain.sh
```

Crie um diretório de construção separado novamente:

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

Prepare o Binutils para compilação:

```
../configure \
  --prefix=/usr \
  --build=$(../config.guess) \
  --host=$LFS_TGT \
  --disable-nls \
  --enable-shared \
  --enable-gprofng=no \
  --disable-werror \
  --enable-64-bit-bfd \
  --enable-new-dtags \
  --enable-default-hash-style=gnu
```

O significado das novas opções de configurar:

--enable-shared
Constrói libbfd como uma biblioteca compartilhada.

--enable-64-bit-bfd
Habilita suporte de 64 bits (em anfitriões com tamanhos de palavra mais estreitos). Isso possivelmente não seja necessário em sistemas de 64 bits, porém não causa dano.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Remova os arquivos de arquivamento da libtool, pois eles são danosos para compilação cruzada e remove bibliotecas estáticas desnecessárias:

```
rm -v $LFS/usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes,sframe}.{a,la}
```

Detalhes deste pacote estão localizados na Seção 8.22.2, “Conteúdo do Binutils.”

6.18. GCC-16.2.0 - Passagem 2

O pacote GCC contém a GNU Compiler Collection, a qual inclui os compiladores C e C++.

Tempo aproximado de 5,2 UPC

construção:

Espaço em disco exigido: 6,4 GB

6.18.1. Instalação do GCC

Como na primeira construção do GCC, os pacotes GMP, MPFR e MPC são exigidos. Desempacote os tarballs e mova-os para os diretórios exigidos:

```
tar -xf ../mpfr-4.2.2.tar.xz
mv -v mpfr-4.2.2 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.4.1.tar.xz
mv -v mpc-1.4.1 mpc
```

Se você estiver construindo em x86_64, mude o nome padrão de diretório para bibliotecas de 64 bits para “lib”:

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

Crie um diretório de construção separado novamente:

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

Antes de iniciar a construção do GCC, lembre-se de desconfigurar quaisquer variáveis de ambiente que substituam os sinalizadores de otimização padrão.

Agora prepare o GCC para compilação:

```
../configure \
  --build=$(../config.guess) \
  --host=$LFS_TGT \
  --target=$LFS_TGT \
  --prefix=/usr \
  --with-build-sysroot=$LFS \
  --enable-default-pie \
  --enable-default-ssp \
  --disable-fixincludes \
  --disable-nls \
  --disable-multilib \
  --disable-libatomic \
  --disable-libgomp \
  --disable-libquadmath \
  --disable-lsanitizer \
  --disable-libssp \
  --disable-libvtv \
  --enable-languages=c,c++ \
  CXX_FOR_TARGET="$LFS_TGT-gcc -nostdinc++" \
  LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc \
  target_configargs=gcc_cv_target_thread_file=posix
```

O significado das novas opções de configurar:

```
--target=$LFS_TGT
```

Nós estamos compilando cruzadamente o GCC, de forma que é impossível construir bibliotecas alvo (libgcc e libstdc++) com os binários do GCC compilados nesta passagem—esses binários não executariam no anfitrião.

O sistema de construção do GCC tentará usar os compiladores C e C++ do anfitrião como uma solução alternativa por padrão. Construir as bibliotecas alvo do GCC com uma versão diferente do GCC não é suportado, de forma que usar os compiladores do anfitrião possivelmente cause falha de construção. Esse parâmetro assegura que as bibliotecas sejam construídas pelo GCC passagem 1.

```
--with-build-sysroot=$LFS
```

Normalmente, usar `--host` garante que um compilador cruzado seja usado para construir o GCC e que o compilador sabe que tem que procurar por cabeçalhos e por bibliotecas em `$LFS`. Porém, o sistema de construção para o GCC usa ferramentas adicionais, que não estão cientes desse local. Essa chave é necessária, de forma que tais ferramentas procurarão pelos arquivos necessários em `$LFS`, e não no anfitrião.

```
--disable-fixincludes
```

Por padrão, durante a instalação do GCC, alguns cabeçalhos do sistema seriam “corrigidos” para serem usados com o GCC. Isso não é necessário para um sistema Linux moderno e potencialmente danoso se um pacote for reinstalado depois da instalação do GCC. Essa chave impede o GCC de “corrigir” os cabeçalhos.

```
--disable-lsanitizer
```

Desabilita as bibliotecas sanitizadoras de tempo de execução do GCC. Elas não são necessárias para a instalação temporária. Em `gcc-pass1` estava implícito por `--disable-libstdc++` e agora temos que passá-lo explicitamente.

```
CXX="$LFS_TGT-gcc -nostdinc++"
```

Especifica o uso de `$LFS_TGT-gcc` para compilar o código C++ como em Seção 5.6, “Libstdc++ oriundo de GCC-16.2.0” e impede que o compilador use os cabeçalhos originários da instalação anterior da `libstdc++`. Misturar-se esses cabeçalhos com os cabeçalhos C++ provenientes da construção atual da `libstdc++` pode fazer com que as diretivas `#include_next` falhem ao localizar os cabeçalhos C. Em uma construção normal, essa definição é passada automaticamente para o conjunto de comandos sequenciados **configure** a partir do diretório de nível superior. Aqui, nós a passamos explicitamente.

```
LDFLAGS_FOR_TARGET=...
```

Permite que `libstdc++` use o `libgcc` sendo construído nessa passagem, em vez da versão anterior construída em `gcc-pass1`. A versão anterior não pode suportar adequadamente o tratamento de exceções C++ porque foi construída sem suporte a `libc`.

```
target_configargs=gcc_cv_target_thread_file=posix
```

Construa as bibliotecas alvo `libgcc` e `libstdc++` com suporte de camada POSIX habilitado. O padrão é o de seguir a configuração do compilador usado para construir a biblioteca alvo (nesse caso, `gcc-pass1` que foi configurado com nenhum suporte a camada).

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS install
```

Como um toque final, crie um link simbólico utilitário. Muitos aplicativos e scripts executam `cc` em vez de `gcc`, o que é usado para manter genéricos os aplicativos e, assim, utilizáveis em todos os tipos de sistemas UNIX onde o compilador GNU C nem sempre está instalado. Executar `cc` deixa o(a) administrador(a) do sistema livre para decidir qual compilador C instalar:

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

Detalhes acerca deste pacote estão localizados na Seção 8.32.2, “Conteúdo do GCC.”

Capítulo 7. Entrando no Chroot e Construindo Ferramentas Temporárias Adicionais

7.1. Introdução

Este capítulo mostra como construir os bits ausentes finais do sistema temporário: as ferramentas necessárias para construir os vários pacotes. Agora que todas as dependências circulares foram resolvidas, um ambiente “chroot”, completamente isolado do sistema operacional anfitrião (exceto pelo núcleo em execução), pode ser usado para a construção.

Para a operação adequada do ambiente isolado, alguma comunicação com o núcleo em execução precisa ser estabelecida. Isso é feito por meio dos assim chamados *Sistemas de Arquivos Virtuais do Núcleo*, que serão montados antes da entrada no ambiente chroot. Você possivelmente queira verificar se eles estão montados emitindo o comando **findmnt**.

Até a Seção 7.4, “Entrando no Ambiente Chroot”, os comandos precisam ser executados como **root**, com a variável **LFS** configurada. Após a entrada no chroot, todos os comandos são executados como **root**, por sorte sem acesso ao SO do computador no qual que você construiu o LFS. Seja cuidadoso(a) de qualquer maneira, dado que é fácil destruir o sistema LFS inteiro com comandos mau formados.

7.2. Mudando Propriedade



Nota

Os comandos no resto deste livro precisam ser realizados enquanto logado(a) como usante **root** e não mais como usante **lfs**. Também, verifique duplamente se **\$LFS** está configurada no ambiente do(a) **root**.

Atualmente, a hierarquia inteira de diretórios em **\$LFS** é titularizada pelo(a) usante **lfs**, um(a) usante que existe somente no sistema anfitrião. Se os diretórios e arquivos sob **\$LFS** forem mantidos como estão, eles serão titularizados por um ID de usante sem uma conta correspondente. Isso é perigoso, pois uma conta de usante criada posteriormente poderia obter esse mesmo ID de usante e titularizaria todos os arquivos sob **\$LFS**, assim expondo esses arquivos a possível manipulação maliciosa.

Para endereçar esse problema, mude a titularidade de propriedade dos diretórios **\$LFS/*** para usante **root** executando o seguinte comando:

```
chown --from lfs -R root:root $LFS/{usr,var,etc,tools}
case $(uname -m) in
    x86_64) chown --from lfs -R root:root $LFS/lib64 ;;
esac
```

7.3. Preparando Sistemas de Arquivos Virtuais do Núcleo

Aplicativos executando no espaço de usante utilizam vários sistemas de arquivos criados pelo núcleo para comunicar com o próprio núcleo. Esses sistemas de arquivos são virtuais: nenhum espaço em disco é usado por eles. O conteúdo desses sistemas de arquivos reside em memória. Esses sistemas de arquivos precisam estar montados na árvore de diretórios **\$LFS**, de modo que os aplicativos consigam encontrá-los no ambiente chroot.

Comece criando os diretórios nos quais esses sistemas de arquivos virtuais serão montados:

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

7.3.1. Montando e Povoando /dev

Durante uma inicialização normal de um sistema LFS, o núcleo automaticamente monta o sistema de arquivos `devtmpfs` no diretório `/dev`; o núcleo cria nós de dispositivo naquele sistema de arquivos virtuais durante o processo de inicialização ou quando um dispositivo for primeiro detectado ou acessado. O processo de segundo plano `udev` possivelmente mude a titularidade da propriedade ou as permissões dos nós de dispositivo criados pelo núcleo e crie novos nós de dispositivo ou linques simbólicos para facilitar o trabalho dos(as) mantenedores(as) de distribuição e de administradores(as) de sistema. (Veja-se o Seção 9.3.2.2, “Criação de Nó de Dispositivo” para detalhes). Se o núcleo do anfitrião suportar `devtmpfs`, nós podemos simplesmente montar um `devtmpfs` em `$LFS/dev` e confiar no núcleo para povoá-lo.

Porém, alguns núcleos de anfitrião carecem de suporte a `devtmpfs`; essas distribuições anfitriãs usam métodos diferentes para criar o conteúdo do `/dev`. Assim, a única maneira agnóstica ao anfitrião para povoar o diretório `$LFS/dev` é a de montar vinculadamente o diretório `/dev` do sistema anfitrião. Uma montagem vinculada é um tipo especial de montagem que produz uma sub árvore de diretórios ou um arquivo visível em algum outro local. Use o seguinte comando para fazer isso.

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

7.3.2. Montando Sistemas de Arquivos Virtuais do Núcleo

Agora monte os restantes sistemas de arquivos virtuais do núcleo:

```
mount -vt devpts devpts -o gid=5,mode=0620 $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

O significado das opções de montagem para `devpts`:

gid=5

Isso garante que todos os nós de dispositivos criados pelo `devpts` sejam titularizados pelo ID de grupo 5. Esse é o ID que nós usaremos mais tarde para o grupo `tty`. Nós usamos o ID de grupo em vez de um nome, pois o sistema anfitrião pode usar um ID diferente para o grupo `tty` dele.

mode=0620

Isso garante que todos os nós de dispositivos criados pelo `devpts` tenham o modo 0620 (legível e escrevível por usuário, escrevível por grupo). Juntamente com a opção acima, isso garante que o `devpts` criará nós de dispositivos que atendam às exigências da `grantpt()`, significando que o binário auxiliar da Glibc **pt_chown** (que não é instalado por padrão) não é necessário.

Em alguns sistemas anfitriões, `/dev/shm` é um linque simbólico para um diretório, tipicamente `/run/shm`. O `tmpfs` do `/run` foi montado acima, de modo que, nesse caso, somente um diretório precisa ser criado com as permissões corretas.

Em outros sistemas anfitriões, `/dev/shm` é um ponto de montagem para um `tmpfs`. Nesse caso, a montagem do `/dev` acima somente criará `/dev/shm` como um diretório no ambiente `chroot`. Nessa situação, nós precisamos montar explicitamente um `tmpfs`:

```
if [ -h $LFS/dev/shm ]; then
    install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
    mount -vt tmpfs -o nosuid,nodev tmpfs $LFS/dev/shm
fi
```

7.4. Entrando no Ambiente Chroot

Agora que todos os pacotes que são exigidos para construir o resto das ferramentas necessárias estão no sistema, é tempo de entrar no ambiente chroot e finalizar a instalação das ferramentas temporárias. Esse ambiente também será usado para instalar o sistema final. Como usante `root`, execute o seguinte comando para entrar no ambiente que é, neste momento, povoado com nada, mas ferramentas temporárias:

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

Se você não quiser usar todos os núcleos lógicos disponíveis, [então] substitua `$(nproc)` pelo número de núcleos lógicos que você deseja usar para construir pacotes neste capítulo e nos capítulos seguintes. As suítes de teste de alguns pacotes (notadamente "Autoconf", "Libtool" e "Tar") no Capítulo 8 não são afetadas por "MAKEFLAGS"; em vez disso, elas usam uma variável de ambiente "TESTSUITEFLAGS". Nós configuramos isso aqui também para executar essas suítes de teste com vários núcleos.

A opção `-i` dada para o comando `env` limpará todas as variáveis no ambiente chroot. Depois disso, somente as variáveis `HOME`, `TERM`, `PS1`, e `PATH` são configuradas novamente. A construção `TERM=$TERM` configura a variável `TERM` dentro do chroot para o mesmo valor que fora do chroot. Essa variável é necessária, de modo que aplicativos como o `vim` e o `less` possam operar adequadamente. Se outras variáveis forem desejadas, tais como `CFLAGS` ou `CXXFLAGS`, [então] esse é um bom lugar para configurá-las.

Deste ponto em diante, não existe mais necessidade de usar a variável `LFS`, pois todo o trabalho estará restrito ao sistema de arquivos do LFS; o comando `chroot` executa o shell Bash com o diretório raiz (`/`) configurado para `$LFS`.

Observe que `/tools/bin` não está no `PATH`. Isso significa que o conjunto de ferramentas cruzadas não mais será usado.

Observe também que o "prompt" do "`bash`" dirá "I have no name!" Isso é normal porque o arquivo `/etc/passwd` ainda não foi criado.



Nota

É importante que todos os comandos até o final deste capítulo e nos capítulos seguintes sejam executados de dentro do ambiente chroot. Se você deixar esse ambiente por qualquer razão (reinicializar, por exemplo), [então] certifique-se de que os sistemas de arquivos virtuais do núcleo estejam montados como explicado no Seção 7.3.1, "Montando e Povoando /dev" e no Seção 7.3.2, "Montando Sistemas de Arquivos Virtuais do Núcleo" e entre no chroot novamente antes de continuar com a instalação.

7.5. Criando Diretórios

É tempo de criar a estrutura completa de diretórios no sistema de arquivos do LFS.



Nota

Alguns dos diretórios mencionados nesta seção possivelmente já tenham sido criados anteriormente com instruções explícitas ou quando da instalação de alguns pacotes. Elas estão repetidas abaixo para completude.

Crie alguns diretórios de nível de raiz que não estão no conjunto limitado exigido nos capítulos anteriores emitindo o seguinte comando:

```
mkdir -pv /{boot,home,mnt,opt,srv}
```

Crie o conjunto exigido de subdiretórios abaixo do nível de raiz emitindo os seguintes comandos:

```
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{include,src}
mkdir -pv /usr/lib/locale
mkdir -pv /usr/local/{bin,lib,sbin}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

Diretórios são, por padrão, criados com modo de permissão 755, mas isso não é desejável em todos os lugares. Nos comandos acima, duas mudanças são feitas—uma para o diretório lar do(a) usante `root` e outra para os diretórios para arquivos temporários.

A primeira mudança de modo assegura que nem toda pessoa consiga entrar no diretório `/root`—exatamente como um(a) usante normal faria com o próprio diretório lar dele ou dela. A segunda mudança de modo garante que qualquer usante consiga escrever nos diretórios `/tmp` e `/var/tmp`, mas não consiga remover deles os arquivos de outros(as) usantes. A última é proibida pelo assim chamado “sticky bit”, o bit mais alto (1) na máscara de bits 1777.

7.5.1. Observação de Conformidade com o FHS

Essa árvore de diretórios é baseada no Padrão de Hierarquia de Sistema de Arquivos (Filesystem Hierarchy Standard - FHS) (disponível em <https://refspecs.linuxfoundation.org/fhs.shtml>). O FHS também especifica a existência opcional de diretórios adicionais, tais como `/usr/local/games` e `/usr/share/games`. No LFS, nós criamos apenas os diretórios que são realmente necessários. Entretanto, sintase livre para criar mais diretórios, se você desejar.



Atenção

O FHS não impõe a existência do diretório `/usr/lib64` e os(as) editores(as) do LFS decidiram não usá-lo. Para as instruções no LFS e no BLFS funcionarem corretamente, é imperativo que esse diretório seja não existente. De tempos em tempos você deveria verificar se ele não existe, pois é fácil criá-lo inadvertidamente e isso provavelmente quebrará o seu sistema.

7.6. Criando Arquivos Essenciais e Linques Simbólicos

Historicamente, o Linux manteve uma lista dos sistemas de arquivos montados no arquivo `/etc/mtab`. Núcleos modernos mantém essa lista internamente e a expõem para o(a) usante via sistema de arquivos `/proc`. Para satisfazer utilitários que esperam encontrar o `/etc/mtab`, crie o seguinte linque simbólico:

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

Crie um arquivo `/etc/hosts` básico para ser referenciado em algumas suítes de teste e em um dos arquivos de configuração do Perl também:

```
cat > /etc/hosts << EOF
127.0.0.1 localhost $(hostname)
::1 localhost
EOF
```

Para que o(a) usante `root` seja capaz de logar e para que o nome “root” seja reconhecido, precisam existir entradas relevantes nos arquivos `/etc/passwd` e `/etc/group`.

Crie o arquivo `/etc/passwd` executando o seguinte comando:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
systemd-journal-gateway:x:73:73:systemd Journal Gateway:/usr/bin/false
systemd-journal-remote:x:74:74:systemd Journal Remote:/usr/bin/false
systemd-journal-upload:x:75:75:systemd Journal Upload:/usr/bin/false
systemd-network:x:76:76:systemd Network Management:/usr/bin/false
systemd-resolve:x:77:77:systemd Resolver:/usr/bin/false
systemd-timesync:x:78:78:systemd Time Synchronization:/usr/bin/false
systemd-coredump:x:79:79:systemd Core Dumper:/usr/bin/false
uidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
systemd-oom:x:81:81:systemd Out Of Memory Daemon:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

A senha atual para root será configurada posteriormente.

Crie o arquivo `/etc/group` executando o seguinte comando:

```
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:
clock:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
kvm:x:61:
systemd-journal-gateway:x:73:
systemd-journal-remote:x:74:
systemd-journal-upload:x:75:
systemd-network:x:76:
systemd-resolve:x:77:
systemd-timesync:x:78:
systemd-coredump:x:79:
uidd:x:80:
systemd-oom:x:81:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

Os grupos criados não são parte de nenhum padrão—eles são grupos decididos em parte pelas exigências da configuração do Udev no Capítulo 9 e em parte pelas convenções comuns empregadas por um número de distribuições Linux existentes. Adicionalmente, algumas suítes de teste dependem de usantes ou grupos específicos(as). A Linux Standard Base (LSB, disponível em <https://refspecs.linuxfoundation.org/lb.shtml>) somente recomenda que, além do grupo `root` com um ID de Grupo (GID) de 0, um grupo `bin` com um GID de 1 esteja presente. O GID de 5 é amplamente usado para o grupo `tty` e o número 5 também é usado no `systemd` para

o sistema de arquivos devpts. Todos os outros nomes de grupo e GIDs podem ser escolhidos livremente pelo(a) administrador(a) do sistema, uma vez que programas bem escritos não dependem de números de GID, mas sim usam o nome do grupo.

O ID 65534 é usado pelo núcleo para NFS e espaços de nome de usante separados para usantes e grupos não mapeados(as) (aqueles(as) existem no servidor NFS ou no espaço de nome de usante ancestral, porém “não existem” na máquina local ou no espaço de nome separado). Nós atribuímos `nobody` e `nogroup` para evitar um ID não nominado. Porém, outras distribuições possivelmente tratem esse ID diferentemente, de modo que qualquer programa portátil não deveria depender dessa atribuição.

Alguns testes no Capítulo 8 precisam de um(a) usante regular. Nós adicionamos esse(a) usante aqui e deletamos essa conta ao final daquele capítulo.

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Para remover o prompt “I have no name!”, inicie um novo shell. Uma vez que os arquivos `/etc/passwd` e `/etc/group` tenham sido criados, a resolução de nome de usante e de nome de grupo agora funcionará:

```
exec /usr/bin/bash --login
```

Os aplicativos **login**, **agetty** e **init** (e outros) usam um número de arquivos de registro para registrar informação, tais como quem esteve logada(o) no sistema e quando. Entretanto, esses aplicativos não escreverão nos arquivos de registro se eles já não existirem. Inicialize os arquivos de registro e dê a eles permissões adequadas:

```
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
```

O arquivo `/var/log/wtmp` registra todos os logins e logouts. O arquivo `/var/log/lastlog` registra quando cada usante se logou pela última vez. O arquivo `/var/log/faillog` registra tentativas falhas de login. O arquivo `/var/log/btmp` registra tentativas inválidas de login.



Nota

O arquivo `/run/utmp` registra os(as) usantes que estejam atualmente logados(as). Esse arquivo é criado dinamicamente quando um(a) usante se loga no sistema.



Nota

Os arquivos `utmp`, `wtmp`, `btmp` e `lastlog` usam inteiros de 32 bits para carimbos de tempo e eles serão fundamentalmente quebrados depois do ano 2038. Muitos pacotes pararam de usá-los e outros pacotes pararão de usá-los. Provavelmente é melhor considerá-los obsoletos.

7.7. Gettext-1.0

O pacote Gettext contém utilitários para internacionalização e localização. Eles permitem que programas sejam compilados com Suporte ao Idioma Nativo (Native Language Support - NLS), habilitando-os a emitir mensagens no idioma nativo do(a) usante.

Tempo aproximado de construção: 1,5 UPC

Espaço em disco exigido: 531 MB

7.7.1. Instalação do Gettext

Para nosso conjunto temporário de ferramentas, nós precisamos somente instalar três aplicativos originários do Gettext.

Prepare o Gettext para compilação:

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

O significado da opção de configure:

--disable-shared

Nós não precisamos instalar quaisquer das bibliotecas compartilhadas do Gettext nesta ocasião, assim não existe necessidade de construí-las.

Compile o pacote:

```
make
```

Instale os aplicativos **msgfmt**, **msgmerge** e **xgettext**:

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

Detalhes acerca deste pacote estão localizados na Seção 8.36.2, “Conteúdo do Gettext.”

7.8. Bison-3.8.2

O pacote Bison contém um gerador de analisador.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 58 MB

7.8.1. Instalação do Bison

Prepare o Bison para compilação:

```
./configure --prefix=/usr \  
            --docdir=/usr/share/doc/bison-3.8.2
```

O significado da nova opção do configure:

```
--docdir=/usr/share/doc/bison-3.8.2
```

Isso diz ao sistema de construção para instalar a documentação do bison em um diretório versionado.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.37.2, “Conteúdo do Bison.”

7.9. Perl-5.44.0

O pacote Perl contém o Practical Extraction and Report Language.

Tempo aproximado de construção: 0,6 UPC

Espaço em disco exigido: 301 MB

7.9.1. Instalação do Perl

Prepare o Perl para compilação:

```
sh Configure -des \
-D prefix=/usr \
-D vendorprefix=/usr \
-D useshrplib \
-D privlib=/usr/lib/perl5/5.44/core_perl \
-D archlib=/usr/lib/perl5/5.44/core_perl \
-D sitelib=/usr/lib/perl5/5.44/site_perl \
-D sitearch=/usr/lib/perl5/5.44/site_perl \
-D vendorlib=/usr/lib/perl5/5.44/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.44/vendor_perl
```

O significado das opções do Configure:

-des
Essa é uma combinação de três opções: *-d* usa padrões para todos os itens; *-e* assegura completamento de todas as tarefas; *-s* silencia saída gerada não essencial.

-D vendorprefix=/usr
Isso garante que **perl** saiba como dizer aos pacotes onde eles deveriam instalar os módulos "Perl" deles.

-D useshrplib
Construa a `libperl`, necessária para alguns módulos "Perl", como uma biblioteca compartilhada, em vez de uma biblioteca estática.

-D privlib, -D archlib, -D sitelib, ...
Essas configurações definem onde o "Perl" procura os módulos instalados. Os(As) editores(as) do LFS optaram por colocá-los em uma estrutura de diretórios baseada na versão PRINCIPAL.SECUNDÁRIA do "Perl" (5.44) que permite atualizar o "Perl" para níveis de remendo mais recentes (o nível de remendo é a última parte separada por pontos na sequência completa de caracteres da versão, como 5.44.0) sem reinstalar todos os módulos.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.46.2, “Conteúdo do Perl.”

7.10. Zlib-1.3.2

O pacote Zlib contém rotinas de compressão e descompressão usadas por alguns aplicativos.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 5,7 MB

7.10.1. Instalação do Zlib

Prepare Zlib para compilação:

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

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Remova uma biblioteca estática inútil:

```
rm -fv /usr/lib/libz.a
```

Detalhes acerca desse pacote estão localizados na Seção 8.6.2, “Conteúdo do Zlib.”

7.11. mpdecimal-4.0.1

O pacote mpdecimal contém bibliotecas rápidas C/C++ para aritmética de ponto flutuante decimal de precisão arbitrária com arredondamento correto.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 3,3 MB

7.11.1. Instalação do mpdecimal

Prepare mpdecimal para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpdecimal-4.0.1
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca desse pacote estão localizados na Seção 8.53.2, “Conteúdo do mpdecimal.”

7.12. Python-3.14.7

O pacote Python 3 contém o ambiente de desenvolvimento do Python. Ele é útil para programação orientada a objeto, escrita de scripts, prototipagem de aplicativos grandes e desenvolvimento de aplicações inteiras. Python é uma linguagem interpretada de computador.

Tempo aproximado de construção: 0,5 UPC

Espaço em disco exigido: 603 MB

7.12.1. Instalação do Python



Nota

Existem dois arquivos de pacotes cujos nomes se iniciam com o prefixo “python”. Aquele a se extrair a partir dele é `Python-3.14.7.tar.xz` (perceba a primeira letra maiúscula).

Prepare o Python para compilação:

```
./configure --prefix=/usr \
            --enable-shared \
            --without-ensurepip \
            --without-static-libpython
```

O significado da opção de configure:

`--enable-shared`

Essa chave impede a instalação de bibliotecas estáticas.

`--without-ensurepip`

Essa chave desabilita o instalador de pacote do Python, o qual não é necessário neste estágio.

`--without-static-libpython`

Essa chave impede instalar uma biblioteca estática grande, porém desnecessária.

Compile o pacote:

```
make
```



Nota

Alguns módulos do Python 3 não podem ser construídos agora, pois as dependências não estão instaladas ainda. Para o módulo `ssl`, uma mensagem `Python requires a OpenSSL 1.1.1 or newer` é gerada. A mensagem deveria ser ignorada. Apenas tenha certeza de que o comando de nível superior **make** não tenha falhado. Os módulos opcionais não são necessários agora e eles serão construídos no Capítulo 8.

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.54.2, “Conteúdo do Python 3.”

7.13. Texinfo-7.3

O pacote Texinfo contém aplicativos para leitura, escrita e conversão de páginas info.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 147 MB

7.13.1. Instalação do Texinfo

Prepare o Texinfo para compilação:

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

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.73.2, “Conteúdo do Texinfo.”

7.14. Util-linux-2.42.2

O pacote Util-linux contém diversos aplicativos utilitários.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 211 MB

7.14.1. Instalação do Util-linux

O FHS recomenda usar o diretório `/var/lib/hwclock` em vez do usual diretório `/etc` como o local para o arquivo `adjtime`. Crie esse diretório com:

```
mkdir -pv /var/lib/hwclock
```

Prepare o Util-linux para compilação:

```
./configure --libdir=/usr/lib \
            --runstatedir=/run \
            --disable-chfn-chsh \
            --disable-login \
            --disable-nologin \
            --disable-su \
            --disable-setpriv \
            --disable-runuser \
            --disable-pylibmount \
            --disable-static \
            --disable-liblastlog2 \
            --without-python \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.42.2
```

O significado das opções do configure:

`ADJTIME_PATH=/var/lib/hwclock/adjtime`

Isso configura o local do arquivo gravando informação acerca do relógio de hardware de acordo com o FHS. Isso não é estritamente necessário para essa ferramenta temporária, porém impede a criação de um arquivo em outro local, o qual não seria sobrescrito ou removido quando da construção do pacote `util-linux` final.

`--libdir=/usr/lib`

Essa chave assegura que os links simbólicos `.so` apontem para o arquivo de biblioteca compartilhada no mesmo diretório (`/usr/lib`) diretamente.

`--disable-*`

Essas chaves evitam avisos acerca de componentes de construção que exigem pacotes que não estão no LFS ou ainda não estão instalados.

`--without-python`

Essa chave desabilita o uso do Python. Ela evita tentar construir ligações desnecessárias.

`runstatedir=/run`

Essa chave configura corretamente o local do soquete usado por `uuidd` e `libuuidd`.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.81.2, “Conteúdo do Util-linux.”

7.15. Organizando e Salvando o Sistema Temporário

7.15.1. Limpando

Primeiro, remova a documentação atualmente instalada para evitar que ela termine no sistema final e para salvar cerca de 35 MB:

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

Segundo, em um sistema moderno Linux, os arquivos libtool.la somente são úteis para a libltdl. Nenhuma biblioteca no LFS é carregada pela libltdl e é sabido que alguns arquivos .la podem causar falhas de pacote do BLFS. Remova tais arquivos agora:

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

O tamanho atual do sistema é agora de cerca de 3 GB, entretanto o diretório /tools não mais é necessário. Ele usa cerca de 1 GB de espaço de disco. Delete ele agora:

```
rm -rf /tools
```

7.15.2. Cópia de segurança

Neste ponto, os aplicativos e bibliotecas essenciais foram criados e o seu sistema LFS atual está em um bom estado. Seu sistema agora pode ser copiado para posterior reuso. Em caso de falhas fatais nos capítulos subsequentes, frequentemente acontece que remover tudo e começar de novo (mais cuidadosamente) é a melhor maneira para recuperar. Infelizmente, todos os arquivos temporários serão removidos, também. Para evitar desperdiçar tempo extra para refazer tudo o que tenha sido feito com sucesso, criar uma cópia de segurança do sistema LFS atual possivelmente se prove útil.



Nota

Todos os passos restantes nesta seção são opcionais. Apesar disso, tão logo você comece a instalar pacotes no Capítulo 8, os arquivos temporários serão sobrescritos. Assim, possivelmente seja uma boa ideia fazer uma cópia de segurança do sistema atual conforme descrito abaixo.

Os passos seguintes são realizados a partir do lado de fora do ambiente chroot. Isso significa que você tem de deixar o ambiente chroot primeiro antes de continuar. A razão para isso é a de conseguir acesso a locais do sistema de arquivos do lado de fora do ambiente chroot para armazenar/ler o arquivamento da cópia de segurança, o qual convém não ser colocado dentro da hierarquia do \$LFS.

Se você decidiu fazer uma cópia de segurança, [então] deixe o ambiente chroot:

```
exit
```



Importante

Todas as instruções seguintes são executadas pelo(a) root no teu sistema anfitrião. Tome cuidado extra acerca dos comandos que você vai executar, uma vez que erros cometidos aqui podem modificar teu sistema anfitrião. Esteja ciente de que a variável de ambiente LFS esteja configurada para usante lfs por padrão, porém possivelmente não esteja configurada para root.

Sempre que comandos forem ser executados por root, tenha certeza de que você configurou LFS.

Isso foi discutido na Seção 2.6, “Configurando a Variável \$LFS e o Umask.”

Antes de fazer uma cópia de segurança, desmonte os sistemas de arquivos virtuais:

```
mountpoint -q $LFS/dev/shm && umount $LFS/dev/shm
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Tenha certeza de que tem pelo menos um (01) GB de espaço livre no disco (os tarballs do fonte serão incluídos no arquivamento da cópia de segurança) no sistema de arquivos contendo o diretório onde você criar o arquivamento da cópia de segurança.

Observe que as instruções abaixo especificam o diretório lar do(a) usante root do sistema anfitrião, o qual tipicamente é encontrado no sistema de arquivos raiz. Substitua \$HOME por um diretório da tua escolha se não quiser ter a cópia de segurança armazenada no diretório lar do(a)root.

Crie o arquivamento da cópia de segurança executando o seguinte comando:



Nota

Por causa de que o arquivamento da cópia de segurança é comprimido, dura um tempo relativamente longo (mais que dez (10) minutos) mesmo em um sistema razoavelmente rápido.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-13.1-systemd.tar.xz .
```



Nota

Se continuar para o capítulo 8, [então] não se esqueça de entrar novamente no ambiente chroot conforme explicado na caixa “Importante” abaixo.

7.15.3. Restauo

No caso de alguns erros terem sido cometidos e você precisar começar de novo, você pode usar essa cópia de segurança para restaurar o sistema e economizar algum tempo de recuperação. Desde que os fontes estão localizados sob \$LFS, eles são incluídos no arquivamento da cópia de segurança também, de forma que não precisam ser transferidos novamente. Após verificar se \$LFS está configurada adequadamente, você consegue restaurar a cópia de segurança executando os seguintes comandos:



Atenção

Os seguintes comandos são extremamente perigosos. Se você executar **rm -rf ./*** como o(a) usante root e você não mudar para o diretório \$LFS ou a variável de ambiente LFS não estiver configurada para o(a) usante root, isso destruirá teu sistema anfitrião inteiro. VOCÊ ESTÁ AVISADO(A).

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-13.1-systemd.tar.xz
```

Novamente, verifique duplamente se o ambiente foi configurado adequadamente e continue construindo o resto do sistema.



Importante

Se você deixou o ambiente chroot para criar uma cópia de segurança ou reiniciar a construção usando um restauro, lembre-se de verificar se os sistemas de arquivos virtuais ainda estão montados (**findmnt | grep \$LFS** deveria mostrar ao menos \$LFS/dev, \$LFS/proc e \$LFS/sys como montados). Se eles não estiverem montados, remonte-os agora conforme descrito na Seção 7.3, “Preparando Sistemas de Arquivos Virtuais do Núcleo” e entre novamente no ambiente chroot (veja-se a Seção 7.4, “Entrando no Ambiente Chroot”) antes de continuar.

Parte IV. Construindo o Sistema LFS

Capítulo 8. Instalando Aplicativos Básicos de Sistema

8.1. Introdução

Neste capítulo, nós começamos a construir o sistema LFS pra valer.

A instalação desse software é direta. Embora em muitos casos as instruções de instalação pudessem ser mais curtas e mais genéricas, nós optamos por fornecer as instruções completas para cada pacote para minimizar as possibilidades de erros. A chave para aprender o que faz um sistema Linux funcionar é a de saber para que cada pacote é usado e porque você (ou o sistema) possivelmente precise dele.

Nós não recomendamos usar otimizações personalizadas. Elas podem fazer com que um aplicativo execute ligeiramente mais rápido, mas elas também possivelmente causem dificuldades de compilação e problemas quando executar o aplicativo. Se um pacote se recusar a compilar com uma otimização personalizada, [então] tente compilá-lo sem otimização e veja se isso corrige o problema. Mesmo se o pacote compilar quando usar uma otimização personalizada, existe o risco de que ele possivelmente tenha sido compilado incorretamente devido às complexas interações entre o código e as ferramentas de construção. Observe também que as opções `-march` e `-mtune` usando valores não especificados no livro não foram testadas. Isso possivelmente cause problemas com os pacotes do conjunto de ferramentas (Binutils, GCC e Glibc). Os pequenos ganhos potenciais alcançados personalizando-se otimizações de compilador frequentemente são superados pelos riscos. Construtoras(es) de primeira vez do LFS são encorajadas(os) a construir sem otimizações personalizadas.

Por outro lado, nós mantemos as otimizações habilitadas pela configuração padrão dos pacotes. Adicionalmente, as vezes, nós explicitamente habilitamos uma configuração otimizada fornecida por um pacote, porém não habilitada por padrão. Os(As) mantenedores(as) do pacote já testaram essas configurações e as consideraram seguras, de modo que não é provável que elas quebrariam a construção. Geralmente, a configuração padrão já habilita `-O2` ou `-O3`, de modo que o sistema resultante ainda executará muito rápido sem qualquer otimização personalizada e será estável ao mesmo tempo.

Antes das instruções de instalação, cada página de instalação fornece informação a respeito do pacote, incluindo uma descrição concisa do que ele contém, aproximadamente quando tempo levará para construir, e quanto espaço de disco é exigido durante esse processo de construção. Seguindo as instruções de instalação, existe uma lista de aplicativos e de bibliotecas (juntamente com breves descrições) que o pacote instala.



Nota

Os valores da UPC e espaço em disco exigido incluem dados da suíte de teste para todos os pacotes aplicáveis no Capítulo 8. Os valores da UPC foram calculados usando quatro núcleos da CPU (`-j4`) para todas as operações, a menos que especificado do contrário.

8.1.1. Acerca de bibliotecas

Em geral, as(os) editoras(es) do LFS desencorajam construir e instalar bibliotecas estáticas. A maior parte das bibliotecas estáticas tem sido tornada obsoleta em um sistema moderno Linux. Além disso, vincular uma biblioteca estática a um aplicativo pode ser prejudicial. Se uma atualização para a biblioteca for necessária para remover um problema de segurança, [então] cada aplicativo que usar a biblioteca estática precisará ser vinculado novamente à nova biblioteca. Como o uso de bibliotecas estáticas nem sempre é óbvio, os aplicativos relevantes (e os procedimentos necessários para fazer a vinculação) possivelmente nem mesmo sejam conhecidos.

Os procedimentos neste capítulo removem ou desabilitam a instalação da maioria das bibliotecas estáticas. Usualmente isso é feito passando-se uma opção `--disable-static` para o **configure**. Em outros casos, meios alternativos são necessários. Em uns poucos casos, especialmente Glibc e GCC, o uso de bibliotecas estáticas permanece um recurso essencial do processo de construção do pacote.

Para uma discussão mais completa acerca de bibliotecas, veja-se *Bibliotecas: Estática ou compartilhada?* no livro BLFS.

8.2. Gerenciamento de Pacote

Gerenciamento de Pacote é uma adição frequentemente solicitada para o Livro LFS. Um Gerenciador de Pacote rastreia a instalação de arquivos, tornando mais fácil remover e atualizar pacotes. Um bom gerenciador de pacote também lidará com os arquivos de configuração, especialmente para manter a configuração do(a) usante quando o pacote for reinstalado ou atualizado. Antes que você comece a questionar, NÃO—esta seção não falará nem recomendará qualquer gerenciador de pacote em particular. O que ela fornece é um resumo acerca das técnicas mais populares e como elas funcionam. O gerenciador de pacote perfeito para você possivelmente esteja entre estas técnicas ou possivelmente seja uma combinação de duas ou mais destas técnicas. Esta seção menciona brevemente problemas que possivelmente surjam ao se atualizar pacotes.

Algumas razões porque nenhum gerenciador de pacote é mencionado no LFS ou no BLFS incluem:

- Lidar com gerenciamento de pacote retira o foco das finalidades desses livros—ensinar como um sistema Linux é construído.
- Existem múltiplas soluções para gerenciamento de pacote, cada uma tendo seus pontos fortes e fracos. Encontrar uma solução que satisfaça todas as audiências é difícil.

Existem algumas dicas escritas no tópico do gerenciamento de pacote. Visite o *Hints Project* e veja se uma delas se adéqua às suas necessidades.

8.2.1. Problemas de Atualização

Um Gerenciador de Pacote torna fácil atualizar para versões mais novas quando elas são liberadas. Geralmente as instruções nos livros LFS e BLFS podem ser usadas para atualizar para as versões mais novas. Aqui estão alguns pontos que você deveria estar ciente quando da atualização de pacotes, especialmente em um sistema em execução.

- Se o núcleo Linux precisar ser atualizado (por exemplo, de 5.10.17 para 5.10.18 ou 5.11.1), nada mais precisará ser reconstruído. O sistema seguirá funcionando bem graças à interface bem definida entre o núcleo e o espaço de usante. Especificamente, os cabeçalhos da API do Linux não precisarão ser atualizados juntamente com o núcleo. Você meramente precisará reinicializar o teu sistema para usar o núcleo atualizado.
- Se a Glibc precisar ser atualizada para uma versão mais recente (por exemplo, da Glibc-2.36 para a Glibc-2.44), [então] algumas etapas extras serão necessárias para evitar quebrar o sistema. Leia-se Seção 8.5, “Glibc-2.44” para detalhes.
- Se um pacote contendo uma biblioteca compartilhada for atualizado e se o nome da biblioteca¹ mudar, então quaisquer pacotes dinamicamente vinculados à biblioteca precisam ser recompilados, para vincular contra a biblioteca mais nova. Por exemplo, considere um pacote `foo-1.2.3` que instala uma biblioteca compartilhada com o nome `libfoo.so.1`. Suponha que você atualize o pacote para uma versão mais nova `foo-1.2.4` que instala uma biblioteca compartilhada com o nome `libfoo.so.2`. Nesse caso, quaisquer pacotes que estiverem dinamicamente vinculados à `libfoo.so.1` precisam ser recompilados para vincular contra `libfoo.so.2` para a finalidade de usar a nova versão da biblioteca. Você não deveria remover as bibliotecas antigas até que todos os pacotes dependentes tenham sido recompilados.
- Se um pacote estiver (direta ou indiretamente) lincado aos nomes antigo e novo de uma biblioteca compartilhada (por exemplo, o pacote linca para `libfoo.so.2` e `libbar.so.1`, enquanto o último se linca a `libfoo.so.3`), o pacote possivelmente funcione mal, pois as diferentes revisões da biblioteca compartilhada apresentam definições incompatíveis para alguns nomes de símbolo. Isso pode ser causado pela recompilação de alguns, mas não todos, dos pacotes lincados à antiga biblioteca compartilhada depois que o pacote que

O nome de uma biblioteca compartilhada é a sequência de caracteres codificada na entrada `DT_SONAME` da seção dinâmica ELF dela. Você consegue obtê-lo com o comando `readelf -d <arquivo da biblioteca> | grep SONAME`. Na maioria dos casos, ele é sufixado com `.so.<um número de versão>`, mas existem alguns casos em que ele contém vários números para versionamento (como `libbz2.so.1.0`); contém o número da versão antes do sufixo `.so` (como `libbfd-2.47`); ou não contém nenhum número de versão (por exemplo, `libmemusage.so`). Geralmente não existe correlação entre a versão do pacote e o(s) número(s) da versão no nome da biblioteca.

fornece a biblioteca compartilhada for atualizado. Para evitar o problema, usantes precisarão reconstruir cada pacote lincado a uma biblioteca compartilhada com uma revisão atualizada (por exemplo, `libfoo.so.2` para `libfoo.so.3`) o mais rápido possível.

- Se um pacote contendo uma biblioteca compartilhada for atualizado e o nome da biblioteca não mudar, porém o número de versão do **arquivo** da biblioteca decrescer (por exemplo, a biblioteca ainda é chamada de `libfoo.so.1`, porém o nome do arquivo da biblioteca for mudado de `libfoo.so.1.25` para `libfoo.so.1.24`), [então] você deveria remover o arquivo da biblioteca originário da versão previamente instalada (`libfoo.so.1.25` nesse caso). Do contrário, um comando **ldconfig** (invocado por você mesmo(a) a partir da linha de comando ou pela instalação de algum pacote) reconfigurará o link simbólico `libfoo.so.1` para apontar para o antigo arquivo da biblioteca, pois ele aparenta ser uma versão “mais nova”; o número de versão dele é mais largo. Essa situação possivelmente surge se você tiver que desatualizar um pacote ou se os(as) autores(as) mudarem o esquema de versionamento para arquivos de biblioteca.
- Se um pacote contendo uma biblioteca compartilhada for atualizado e o nome da biblioteca não mudar, porém um problema severo (especialmente, uma vulnerabilidade de segurança) for corrigido, [então] todos os aplicativos em execução vinculados à biblioteca compartilhada deveriam ser reiniciados. O seguinte comando, executado como `root` depois que a atualização estiver completa, listará quais processos estão usando as versões antigas daquelas bibliotecas (substitua `libfoo` pelo nome da biblioteca):

```
grep -l 'libfoo.*deleted' /proc/*/maps | tr -cd 0-9\\n | xargs -r ps u
```

Se o OpenSSH estiver sendo usado para acessar o sistema e ele estiver vinculado à biblioteca atualizada, [então] você precisa reiniciar o serviço **sshd**, então deslogar-se, logar-se novamente e executar o comando precedente novamente para confirmar se nada ainda está usando as bibliotecas deletadas.

Se o processo de segundo plano do **systemd** (executando como PID 1) for lincado à biblioteca atualizada, você pode reiniciá-lo sem reinicializar executando **systemctl daemon-reexec** como o(a) usante `root`.

- Se um aplicativo executável ou uma biblioteca compartilhada for sobrescrito, [então] os processos usando o código ou os dados naquele aplicativo ou biblioteca possivelmente quebrem. A maneira correta para atualizar um aplicativo ou uma biblioteca compartilhada sem causar quebra ao processo é a de removê-lo primeiro, então instalar a versão nova. O comando **install** fornecido por `coreutils` já implementou isso e a maioria dos pacotes usa esse comando para instalar arquivos binários e bibliotecas. Isso significa que você não estaria encrencada(o) por esse problema a maior parte do tempo. Entretanto, o processo de instalação de alguns pacotes (notadamente "SpiderMonkey" no BLFS) apenas sobrescreve o arquivo se ele existir; isso causa uma quebra. Assim, é mais seguro salvar seu trabalho e fechar processos em execução desnecessários antes de atualizar um pacote.

8.2.2. Técnicas de Gerenciamento de Pacote

As seguintes são algumas técnicas comuns de gerenciamento de pacote. Antes de se decidir acerca de um gerenciador de pacote, pesquise sobre as várias técnicas, particularmente os pontos fracos de cada esquema em particular.

8.2.2.1. Está Tudo na Minha Cabeça!

Sim, essa é uma técnica de gerenciamento de pacote. Algumas pessoas não necessitam de um gerenciador de pacote, pois elas conhecem os pacotes intimamente e sabem quais arquivos estão instalados por cada pacote. Alguns(mas) usantes também não precisam de qualquer gerenciamento de pacote, pois eles(as) planejam reconstruir o sistema inteiro sempre que um pacote for mudado.

8.2.2.2. Instalação em Diretórios Separados

Essa é uma técnica simplista de gerenciamento de pacotes que não precisa de um aplicativo especial para gerenciar os pacotes. Cada pacote é instalado em um diretório separado. Por exemplo, o pacote "foo-1.1" é instalado em `/opt/foo-1.1` e um link simbólico é feito de `/opt/foo` para `/opt/foo-1.1`. Quando uma nova versão "foo-1.2" surge, ela é instalada em `/opt/foo-1.2` e o link simbólico anterior é substituído por um link simbólico para a nova versão.

Variáveis de ambiente, como "PATH", "MANPATH", "INFOPATH", "PKG_CONFIG_PATH", "CPPFLAGS", "LDFLAGS", e o arquivo de configuração "/etc/ld.so.conf", possivelmente precisam ser expandidas para incluir os subdiretórios correspondentes em "/opt/foo -x.y".

Esse esquema é usado pelo livro BLFS para instalar alguns pacotes muito grandes para facilitar a atualização deles. Se você instalar mais que alguns pacotes, [então] esse esquema se tornará ingerenciável. E alguns pacotes (por exemplo, os cabeçalhos de "API" do "Linux" e "Glibc") possivelmente não funcionem bem com esse esquema. **Nunca use esse esquema em todo o sistema.**

8.2.2.3. Gerenciamento de Pacote Estilo Link Simbólico

Essa é uma variação da técnica anterior de gerenciamento de pacote. Cada pacote é instalado como no esquema anterior. Mas, em vez de fazer o linque simbólico via um nome genérico de pacote, cada arquivo é simbolicamente lincado à hierarquia /usr. Isso remove a necessidade de expandir as variáveis de ambiente. Ainda que os linques simbólicos possam ser criados pelo(a) usante, muitos gerenciadores de pacote usam essa abordagem e automatizam a criação dos linques simbólicos. Alguns dos populares inclui Stow, Epkg, Graft, e Depot.

O script de instalação precisa ser falseado, de modo que o pacote pense que está instalado em /usr, ainda que, na realidade, ele esteja instalado na hierarquia /usr/pkg. Instalar dessa maneira geralmente não é uma tarefa trivial. Por exemplo, suponha que você está instalando um pacote libfoo-1.1. As seguintes instruções possivelmente não instalem adequadamente o pacote:

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

A instalação funcionará, mas os pacotes dependentes possivelmente não se vinculem à libfoo conforme você esperaria. Se você compilar um pacote que se vincula contra a libfoo, [então] você possivelmente perceba que ele está vinculado a /usr/pkg/libfoo/1.1/lib/libfoo.so.1 em vez de /usr/lib/libfoo.so.1 como você esperaria. A abordagem correta é a de usar a variável DESTDIR para direcionar a instalação. Essa abordagem funciona como se segue:

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

A maioria dos pacotes suporta essa abordagem, mas existem alguns que não. Para os pacotes não conformes, você possivelmente ou precise instalar manualmente o pacote, ou você possivelmente ache que é mais fácil instalar alguns pacotes problemáticos em /opt.

8.2.2.4. Baseado em Carimbo de Tempo

Nessa técnica, um arquivo é carimbado temporalmente antes da instalação do pacote. Depois da instalação, um simples uso do comando **find** com as opções apropriadas consegue gerar um registro de todos os arquivos instalados após o arquivo de carimbo de tempo ser criado. Um gerenciador de pacote que use essa abordagem é instalação-registro.

Ainda que esse esquema tenha a vantagem de ser simples, ele tem duas desvantagens. Se, durante a instalação, os arquivos forem instalados com algum outro carimbo de tempo que não a hora atual, [então] aqueles arquivos não serão rastreados pelo gerenciador de pacote. Além disso, esse esquema pode ser usado apenas quando um pacote for instalado de cada vez. Os registros não são confiáveis se dois pacotes forem instalados simultaneamente a partir de dois consoles.

8.2.2.5. Scripts de Rastreamento de Instalação

Nessa abordagem, os comandos que os scripts de instalação realizam são gravados. Existem duas técnicas que se pode usar:

A variável de ambiente `LD_PRELOAD` pode ser configurada para apontar para uma biblioteca a ser pré-carregada antes da instalação. Durante a instalação, essa biblioteca rastreia os pacotes que estão sendo instalados anexando-se a vários executáveis, tais como o **cp**, o **install** e o **mv**, e rastreando as chamadas de sistema que modificam o sistema de arquivos. Para essa abordagem funcionar, todos os executáveis precisam ser dinamicamente vinculados sem o bit `suid` ou `sgid`. Pré-carregar a biblioteca possivelmente cause alguns efeitos colaterais indesejados durante a instalação. Portanto, é uma boa ideia realizar alguns testes para garantir que o gerenciador de pacote não quebre nada e que ele registre todos os arquivos adequados.

Outra técnica é a de usar o **strace**, que registra todas as chamadas de sistema feitas durante a execução dos scripts de instalação.

8.2.2.6. Criando Arquivamentos de Pacote

Nesse esquema, a instalação do pacote é falseada em uma árvore separada como descrito previamente na seção do gerenciamento de pacote estilo Link Simbólico. Depois da instalação, um arquivamento de pacote é criado usando os arquivos instalados. Esse arquivamento é então usado para instalar o pacote na máquina local ou até mesmo em outras máquinas.

Essa abordagem é a usada pela maioria dos gerenciadores de pacote encontrados nas distribuições comerciais. Exemplos de gerenciadores de pacote que seguem essa abordagem são RPM (o qual, incidentalmente, é exigido pela *Linux Standard Base Specification*), `pkg-utils`, `apt` do Debian, e sistema Portage do Gentoo. Uma dica descrevendo como adotar esse estilo de gerenciamento de pacote para sistemas LFS está localizada em <https://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt>.

A criação de arquivos de pacote que incluem informação de dependência é complexa e além do escopo do LFS.

O Slackware usa um sistema baseado em **tar** para arquivamentos de pacote. Esse sistema intencionalmente não manuseia dependências de pacote como gerenciadores de pacote mais complexos fazem. Para detalhes de gerenciamento de pacote do Slackware, veja-se <http://www.slackbook.org/html/package-management.html>.

8.2.2.7. Gerenciamento Baseado em Usante

Esse esquema, único para LFS, foi concebido por Matthias Benkmann e está disponível a partir do *Projetos de Dicas*. Nesse esquema, cada pacote é instalado como um(a) usante separado(a) nos locais padrão. Arquivos pertencentes a um pacote são facilmente identificados checando-se o ID de usante. Os recursos e deficiências dessa abordagem são muito complexos para se descrever nesta seção. Para os detalhes, por favor veja-se a dica em https://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt.

8.2.3. Implantando o LFS em Múltiplos Sistemas

Uma das vantagens de um sistema LFS é a de que não existem arquivos que dependam da posição de arquivos em um sistema de disco. Clonar uma construção do LFS para outro computador com a mesma arquitetura que a do sistema base é tão simples quanto usar **tar** na partição do LFS que contém o diretório raiz (cerca de 900MB descomprimido para uma construção básica do LFS), copiando aquele arquivo via transferência de rede ou CD-ROM/stick USB para o novo sistema e expandindo-o. Depois disso, uns poucos arquivos de configuração terão que ser mudados. Arquivos de configuração que possivelmente precisem ser atualizados incluem: `/etc/hosts`, `/etc/fstab`, `/etc/passwd`, `/etc/group`, `/etc/shadow` e `/etc/ld.so.conf`.

Um núcleo personalizado possivelmente seja necessário para o novo sistema, dependendo das diferenças no hardware do sistema e a configuração original do núcleo.



Importante

Se você quiser implementar o sistema LFS em um sistema com uma CPU diferente, ao construir Seção 8.23, “GMP-6.3.0” e Seção 8.51, “Libffi-3.8.0” você precisa seguir as observações acerca da substituição da otimização específica da arquitetura para produzir bibliotecas adequadas tanto para o sistema anfitrião quanto para o(s) sistema(s) onde você implementará o sistema LFS. Caso contrário, você obterá erros `Illegal Instruction` ao executar o LFS.

O sistema de construção do GMP armazena a opção de otimização específica da arquitetura usada para construir o GMP em `gmp.h`, e o sistema de construção de algum pacote que use GMP consegue lê-la a partir do cabeçalho e usá-la ao construir o próprio pacote. Pelo menos o sistema de construção do MPFR é conhecido por fazer isso. Portanto, simplesmente reconstruir o GMP em um sistema LFS completo não é suficiente: você precisará recompilar o MPFR e talvez outros pacotes que usem GMP se quiser “converter” um sistema LFS completo para ser usado para uma CPU diferente.

Finalmente, o novo sistema tem de ser tornado inicializável via Seção 10.4, “Usando o GRUB para Configurar o Processo de Inicialização”.

8.3. Man-pages-6.18

O pacote Man-pages contém mais que 2.400 páginas de manual.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 54 MB

8.3.1. Instalação das Páginas de Manual

Remova duas páginas de manual para funções de resumo de senha. Libxcrypt fornecerá uma versão melhor dessas páginas de manual:

```
rm -v man3/crypt*
```

Instale as Páginas de Manual executando:

```
make -R GIT=false prefix=/usr install
```

O significado das opções:

-R

Isso impede que **make** configure quaisquer variáveis internas. O sistema de construção de páginas de manual não funciona bem com variáveis internas, mas atualmente não existe como desabilitá-las, exceto passando-se *-R* explicitamente pela linha de comando.

GIT=false

Isso evita que o sistema de construção emita muitas linhas de aviso `git: command not found`.

8.3.2. Conteúdo das Páginas de Manual

Arquivos instalados: várias páginas de manual

Descrições Curtas

man pages	Descreve funções da linguagem de programação C, arquivos importantes de dispositivo e arquivos significantes de configuração
-----------	--

8.4. Iana-Etc-20260805

O pacote Iana-Etc fornece dados para serviços e protocolos de rede de comunicação.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 4,8 MB

8.4.1. Instalação do Iana-Etc

Para esse pacote, nós precisamos somente copiar os arquivos para o lugar:

```
cp -v services protocols /etc
```

8.4.2. Conteúdo do Iana-Etc

Arquivos instalados: /etc/protocols e /etc/services

Descrições Curtas

/etc/protocols	Descreve os vários protocolos DARPA da Internet que estão disponíveis a partir do subsistema TCP/IP
/etc/services	Fornece um mapeamento entre nomes amigáveis textuais para serviços de internet e os não expostos números de porta atribuídos e tipos de protocolos deles

8.5. Glibc-2.44

O pacote Glibc contém a principal biblioteca C. Essa biblioteca fornece as rotinas básicas para alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos, manuseio de sequências de caracteres, correspondência de padrões, aritmética, e daí por diante.

Tempo aproximado de construção: 11 UPC

Espaço em disco exigido: 3,7 GB

8.5.1. Instalação da Glibc

Alguns dos aplicativos Glibc usam o diretório não conforme com o FHS `/var/db` para armazenar os dados em tempo de execução deles. Aplique o seguinte remendo para fazer com que tais aplicativos armazenem os dados em tempo de execução deles nos locais conformes com o FHS:

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

Corrija um problema que faz com que a função `tanh(3)` quebre em alguns processadores x86_64 antigos e corrija instalar com múltiplas tarefas do **make**:

```
patch -Np1 -i ../glibc-2.44-upstream_fixes-1.patch
```

A documentação da Glibc recomenda construir a Glibc em um diretório dedicado à construção:

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

Prepare a Glibc para compilação:

```
../configure --prefix=/usr          \
              --disable-werror      \
              --disable-nscd        \
              libc_cv_slibdir=/usr/lib \
              --enable-stack-protector=strong \
              --enable-kernel=5.10
```

O significado das opções do `configure`:

`--disable-werror`

Essa opção desabilita a opção `-Werror` passada para o GCC. Isso é necessário para executar a suíte de teste.

`--enable-kernel=5.10`

Essa opção diz ao sistema de construção que esta Glibc possivelmente seja usada com núcleos tão antigos quanto 5.10. Isso significa que a geração de contornos, no caso de uma chamada de sistema introduzida em uma versão posterior, não pode ser usada.

`--enable-stack-protector=strong`

Essa opção aumenta a segurança do sistema adicionando código extra para verificar estouros de buffer, como ataques de esmagamento de pilha. Observe que a Glibc sempre substitui explicitamente o padrão do GCC, de forma que essa opção ainda é necessária mesmo que já tenhamos especificado `--enable-default-ssp` para GCC.

`--disable-nscd`

Não construa o processo de segundo plano de armazenamento temporário do serviço de nomes, o qual não mais é usado.

`libc_cv_slibdir=/usr/lib`

Essa variável configura a biblioteca correta para todos os sistemas. Nós não queremos que a `lib64` seja usada.

Compile o pacote:

```
make
```



Importante

Nesta seção, a suíte de teste para a Glibc é considerada crítica. Não pule-a sob qualquer circunstância.

Geralmente uns poucos testes não passam. As falhas de teste listadas abaixo usualmente são seguras ignorar.

```
make check
```

Você possivelmente veja algumas falhas de teste. A suíte de teste da Glibc é de alguma forma dependente do sistema anfitrião. Umas poucas falhas saídas de mais que 6.000 testes geralmente podem ignoradas. Esta é uma lista dos problemas mais comuns vistos para versões recentes do LFS:

- *io/tst-lchmod* é conhecido por falhar no ambiente chroot do LFS.
- Alguns testes, por exemplo *nss/tst-nss-files-hosts-multi* e *nptl/tst-thread-affinity** são conhecidos por falharem devido a um tempo limite (especialmente quando o sistema está relativamente lento e (ou) executando a suíte de teste com várias tarefas make paralelas). Esses testes podem ser identificados com:

```
grep "Timed out" $(find -name \*.out)
```

É possível reexecutar um teste com tempo limite aumentado com **TIMEOUTFACTOR=<fator> make test t=<nome do teste>**. Por exemplo, **TIMEOUTFACTOR=10 make test t=nss/tst-nss-files-hosts-multi** reexecutará *nss/tst-nss-files-hosts-multi* com dez vezes o tempo limite original.

- Adicionalmente, alguns testes possivelmente falhem com um modelo de CPU relativamente antigo (por exemplo *elf/tst-cpu-features-cpuinfo*) ou versão do núcleo do anfitrião (por exemplo *stdlib/tst-arc4random-thread*), ou com um núcleo do anfitrião mais recente que 7.1.8.

Mesmo sendo uma mensagem inofensiva, o estágio de instalação da Glibc reclamará acerca da ausência do `/etc/ld.so.conf`. Evite esse aviso com:

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

Corrija o Makefile para pular uma verificação de sanidade desatualizada que falha com uma configuração moderna da Glibc:

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



Importante

Se atualizar a Glibc para uma nova versão secundária (por exemplo, da Glibc-2.36 para a Glibc-2.44) em um sistema LFS em execução, [então] você precisará tomar algumas precauções extras para evitar quebrar o sistema:

- Atualizar a Glibc em um sistema LFS anterior ao 11.0 (exclusivo) não é suportado. Reconstrua o LFS se você estiver executando um sistema LFS antigo, mas precisar de uma Glibc mais recente.
- Se atualizar em um sistema LFS anterior a 12.0 (exclusivo), [então] instale a Libxcrypt seguindo a Seção 8.29, “Libxcrypt-4.5.2.” Além de uma instalação normal da Libxcrypt, **você DEVE seguir a observação na seção Libxcrypt para instalar a libcrypt.so.1* (substituindo libcrypt.so.1 originária da instalação anterior da Glibc).**
- Se atualizar em um sistema LFS anterior ao 12.1 (exclusivo), [então] remova o aplicativo **nscd**:

```
rm -f /usr/sbin/nscd
```

Se esse sistema (anterior ao LFS 12.1, exclusivo) for baseado em Systemd, [então] também é necessário desabilitar e parar o serviço **nscd** agora:

```
systemctl disable --now nscd
```

- Atualize o núcleo e reinicialize se ele for mais antigo que 5.10 (verifique a versão atual com **uname -r**) ou se quiser atualizá-lo mesmo assim, seguindo a Seção 10.3, “Linux-7.1.8.”
- Atualize os cabeçalhos da API do núcleo se forem mais antigos que 5.10 (verifique a versão atual com **cat /usr/include/linux/version.h**) ou se quiser atualizá-los mesmo assim, seguindo a Seção 5.4, “Cabeçalhos da API do Linux-7.1.8” (mas removendo \$LFS do comando **cp**).
- Realize uma instalação **DESTDIR** e atualize as bibliotecas compartilhadas da Glibc no sistema usando um comando simples **install**:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/*.so.* /usr/lib
```

É obrigatório seguir rigorosamente essas etapas acima, a menos que você entenda completamente o que está fazendo. **Qualquer desvio inesperado possivelmente torne o sistema completamente desusável. VOCÊ ESTÁ AVISADO(A).**

Em seguida, continue a executar o comando **make install**, o comando **sed** contra o **/usr/bin/ldd** e os comandos para instalar as localidades. Assim que estiverem finalizados, reinicialize o sistema imediatamente.

Quando o sistema tiver reinicializado com sucesso, se você estiver executando um sistema LFS anterior à 12.0 (exclusivo) onde o GCC não foi construído com a opção **--disable-fixincludes**, mova dois cabeçalhos do GCC para um local melhor e remova as cópias “corrigidas” obsoletas dos cabeçalhos da Glibc:

```
DIR=$(dirname $(gcc -print-libgcc-file-name))
[ -e $DIR/include/limits.h ] || mv $DIR/include{-fixed,}/limits.h
[ -e $DIR/include/syslimits.h ] || mv $DIR/include{-fixed,}/syslimits.h
rm -rfv $DIR/include-fixed/*
unset DIR
```

Instale o pacote:

```
make install
```

Corrija um caminho codificado rigidamente para o carregador de executável no script **ldd**:

```
sed 's@/RTLDLIST=/s@/usr@@g' -i /usr/bin/ldd
```

Em seguida, instale as localidades que podem fazer o sistema responder em um idioma diferente. Nenhuma dessas localidades é exigida, mas se algumas delas estiverem ausentes, as suítes de teste de alguns pacotes pularão importantes casos de teste.

Localidades individuais podem ser instaladas usando-se o programa **localedef**. Por exemplo, o segundo comando **localedef** abaixo combina a definição de localidade independente do conjunto de caracteres `/usr/share/i18n/locales/cs_CZ` com a definição de mapa de caracteres `/usr/share/i18n/charmaps/UTF-8.gz` e adiciona o resultado ao arquivo `/usr/lib/locale/locale-archive`. As seguintes instruções instalarão o conjunto mínimo de localidades necessário para a cobertura ótima de testes:

```
localedef -i C -f UTF-8 C.UTF-8
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 ISO-8859-1 en_GB
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_ES -f ISO-8859-15 es_ES@euro
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 is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
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 UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.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
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

Adicionalmente, instale a localidade para teu próprio país, idioma e conjunto de caracteres.

Alternativamente, instale todas as localidades listadas no arquivo `glibc-2.44/localedata/SUPPORTED` (inclui cada localidade listada acima e muitas mais) de uma vez com o seguinte comando consumidor de tempo:

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



Nota

A Glibc agora usa a `libidn2` quando resolver nomes internacionalizados de domínio. Essa é uma dependência de tempo de execução. Se essa capacidade for necessária, as instruções para instalar a `libidn2` estão na *página da libidn2 do BLFS*.

8.5.2. Configurando a Glibc

8.5.2.1. Adicionando o nsswitch.conf

O arquivo `/etc/nsswitch.conf` precisa ser criado, pois os padrões da Glibc não funcionam bem em um ambiente de rede de comunicação.

Crie um novo arquivo `/etc/nsswitch.conf` executando o seguinte:

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

passwd: files systemd
group: files systemd
shadow: files systemd

hosts: mymachines resolve [!UNAVAIL=return] files myhostname dns
networks: files

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

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

8.5.2.2. Adicionando Dados de Fuso Horário

Instale e configure os dados de fuso horário com o seguinte:

```
tar -xf ../../tzdata2026c.tar.gz

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

for tz in etcetera southamerica northamerica europe africa antarctica \
    asia australasia backward; 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 tz
```

O significado dos comandos `zic`:

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

Isso cria fusos horários posix sem quaisquer segundos bissextos. É convencional colocá-los em ambos `zoneinfo` e `zoneinfo/posix`. É necessário colocar os fusos horários POSIX em `zoneinfo`, do contrário várias suítes de teste reportarão erros. Em um sistema embarcado, onde o espaço é apertado e você não pretende nunca atualizar os fusos horários, você poderia economizar 1,9 MB não usando o diretório `posix`, mas alguns aplicativos ou suítes de teste poderiam produzir algumas falhas.

```
zic -L leapseconds ...
```

Isso cria fusos horários corretos, incluindo segundos bissextos. Em um sistema embarcado, onde o espaço é apertado e você não pretende nunca atualizar os fusos horários ou se importa com a hora correta, você poderia economizar 1,9 MB omitindo o diretório `right`.

```
zic ... -p ...
```

Isso cria o arquivo `posixrules`. Nós usamos New York, pois POSIX exige que as regras de horário de verão estejam de acordo com regras dos Estados Unidos da América do Norte.

Uma maneira para determinar o fuso horário local é a de executar o seguinte conjunto de comandos sequenciais:

```
tzselect
```

Depois de responder à umas poucas perguntas a respeito do local, o conjunto de comandos sequenciais retornará o nome do fuso horário (por exemplo, *America/Edmonton*). Existem também alguns outros possíveis fusos horários listados em `/usr/share/zoneinfo`, tais como *Canada/Eastern* ou *EST5EDT* que não são identificados pelo conjunto de comandos sequenciais, mas podem ser usados.

Então crie o arquivo `/etc/localtime` executando:

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

Substitua `<xxx>` pelo nome do fuso horário selecionado (por exemplo, *Canada/Eastern*).

8.5.2.3. Configurando o Carregador Dinâmico

Por padrão, o carregador dinâmico (`/lib/ld-linux.so.2`) procura em `/usr/lib` por bibliotecas dinâmicas que são necessárias para aplicativos assim que são executados. Entretanto, se existirem bibliotecas em outros diretórios diferentes do `/usr/lib`, esses precisam ser adicionados ao arquivo `/etc/ld.so.conf` para a finalidade de que o carregador dinâmico encontre elas. Dois diretórios que são comumente conhecidos por conterem bibliotecas adicionais são `/usr/local/lib` e `/opt/lib`; então adicione esses diretórios ao caminho de busca do carregador dinâmico.

Crie um novo arquivo `/etc/ld.so.conf` executando o seguinte:

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

EOF
```

Se desejado, o carregador dinâmico também pode pesquisar um diretório e incluir o conteúdo de arquivos encontrados lá. Geralmente os arquivos nesse diretório incluem uma linha especificando o caminho de biblioteca desejado. Para adicionar essa capacidade, execute os seguintes comandos:

```
cat >> /etc/ld.so.conf << "EOF"
# Adiciona um diretório de inclusão
include /etc/ld.so.conf.d/*.conf

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

8.5.3. Conteúdo do Glibc

Programas instalados:	gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (link simbólico para <code>ld-linux-x86-64.so.2</code> ou <code>ld-linux.so.2</code>), locale, localedef, makedb, mtrace, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump e zic
Bibliotecas instaladas:	<code>ld-linux-x86-64.so.2</code> , <code>ld-linux.so.2</code> , <code>libBrokenLocale.{a,so}</code> , <code>libanl.{a,so}</code> , <code>libc.{a,so}</code> , <code>libc_nonshared.a</code> , <code>libc_malloc_debug.so</code> , <code>libdl.{a,so.2}</code> , <code>libg.a</code> , <code>libm.{a,so}</code> , <code>libmcheck.a</code> , <code>libmemusage.so</code> , <code>libmvec.{a,so}</code> , <code>libnsl.so.1</code> , <code>libnss_compat.so</code> , <code>libnss_dns.so</code> , <code>libnss_files.so</code> , <code>libnss_hesiod.so</code> , <code>libpcprofile.so</code> , <code>libpthread.{a,so.0}</code> , <code>libresolv.{a,so}</code> , <code>librt.{a,so.1}</code> , <code>libthread_db.so</code> e <code>libutil.{a,so.1}</code>
Diretórios instalados:	<code>/usr/include/arpa</code> , <code>/usr/include/bits</code> , <code>/usr/include/gnu</code> , <code>/usr/include/net</code> , <code>/usr/include/netash</code> , <code>/usr/include/netatalk</code> , <code>/usr/include/netax25</code> , <code>/usr/include/neteconet</code> , <code>/usr/include/netinet</code> , <code>/usr/include/netipx</code> , <code>/usr/include/netiucv</code> , <code>/usr/include/netpacket</code> , <code>/usr/include/netrom</code> , <code>/usr/include/netrose</code> , <code>/usr/include/nfs</code> , <code>/usr/include/protocols</code> , <code>/usr/include/rpc</code> , <code>/usr/include/sys</code> , <code>/usr/lib/audit</code> , <code>/usr/lib/gconv</code> , <code>/usr/lib/locale</code> , <code>/usr/libexec/getconf</code> , <code>/usr/share/i18n</code> , <code>/usr/share/zoneinfo</code> e <code>/var/lib/nss_db</code>

Descrições Curtas

gencat	Gera catálogos de mensagem
getconf	Exibe os valores de configuração de sistema para variáveis específicas do sistema de arquivos
getent	Obtém entradas a partir de uma base de dados administrativa
iconv	Realiza conversão de conjuntos de caracteres
iconvconfig	Cria arquivos de configuração de módulos de carregamento rápido do iconv
ldconfig	Configura as ligações de tempo de execução do vinculador dinâmico
ldd	Reporta quais bibliotecas compartilhadas são exigidas por cada dado aplicativo ou biblioteca compartilhada
lddlibc4	Auxilia o ldd com arquivos objeto. Isso não existe em arquiteturas mais novas como x86_64
locale	Imprime várias informações a respeito da localidade atual
localedef	Compila especificações de localidade
makedb	Cria um banco de dados simples a partir de uma entrada gerada textual
mtrace	Lê e interpreta um arquivo de rastreamento de memória e exibe um resumo em formato legível por humanos
pcprofiledump	Despeja informação gerada pelos perfis do PC
pldd	Lista objetos dinâmicos compartilhados usados por processos em execução
sln	Um aplicativo ln vinculado estaticamente
sotruss	Rastreia chamadas de procedimentos de bibliotecas compartilhadas de um comando especificado
sprof	Lê e exibe dados de perfil de objetos compartilhados
tzselect	Pergunta para o(a) usante a respeito do local do sistema e informa a correspondente descrição de fuso horário
xtrace	Rastreia a execução de um aplicativo exibindo a função atualmente executada
zdump	O despejador de fuso horário
zic	O compilador de fuso horário
<code>ld-*.so</code>	O aplicativo auxiliar para executáveis de bibliotecas compartilhadas
<code>libBrokenLocale</code>	Usado internamente pela Glibc como um hack grosseiro para executar aplicativos quebrados (por exemplo, alguns aplicativos Motif). Vejam-se comentários em <code>glibc-2.44/locale/broken_cur_max.c</code> para mais informação
<code>libanl</code>	Biblioteca fictícia que não contém funções. Anteriormente era a biblioteca assíncrona de pesquisa de nome, cujas funções agora estão em <code>libc</code>
<code>libc</code>	A biblioteca principal C
<code>libc_malloc_debug</code>	Liga verificação de alocação de memória quando pré-carregada
<code>libdl</code>	Biblioteca fictícia que não contém funções. Anteriormente era a biblioteca de interface do vinculador dinâmico, cujas funções agora estão em <code>libc</code>
<code>libg</code>	Biblioteca fictícia que não contém funções. Anteriormente era uma biblioteca de tempo de execução para g++
<code>libm</code>	A biblioteca matemática

<code>libmvec</code>	A biblioteca de vetor matemático, vinculada conforme necessária quando <code>libm</code> for usada
<code>libmcheck</code>	Liga verificação de alocação de memória quando vinculada para
<code>libmemusage</code>	Usado por memusage para ajudar a coletar informação a respeito do uso de memória de um aplicativo
<code>libnsl</code>	A biblioteca de serviços de rede de comunicação, agora obsoleta
<code>libnss_*</code>	Os módulos de Name Service Switch, contendo funções para resolução de nomes de dispositivos, nomes de usantes, nomes de grupos, pseudônimos, serviços, protocolos, entre outras coisas. Carregados pela <code>libc</code> conforme a configuração em <code>/etc/nsswitch.conf</code>
<code>libpcprofile</code>	Pode ser pré-carregada para PC perfilar um executável
<code>libpthread</code>	Biblioteca fictícia que não contém funções. Anteriormente continha funções fornecendo a maioria das interfaces especificadas pelas Extensões de Camadas POSIX.1c e as interfaces de semáforos especificadas pelas Extensões de Tempo Real POSIX.1b; agora as funções estão em <code>libc</code>
<code>libresolv</code>	Contém funções para criação, envio e interpretação de pacotes para os servidores de nomes de domínio da Internet
<code>librt</code>	Contém funções fornecendo a maior parte das interfaces especificadas pelas Extensões de Tempo Real POSIX.1b
<code>libthread_db</code>	Contém funções úteis para construir depuradores para aplicativos de múltiplas camadas
<code>libutil</code>	Biblioteca fictícia que não contém funções. Anteriormente continha código para funções “padrão” usadas em muitos utilitários Unix. Essas funções agora estão na <code>libc</code>

8.6. Zlib-1.3.2

O pacote Zlib contém rotinas de compressão e descompressão usadas por alguns aplicativos.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 5,4 MB

8.6.1. Instalação do Zlib

Prepare Zlib para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

Remova uma biblioteca estática inútil:

```
rm -fv /usr/lib/libz.a
```

8.6.2. Conteúdo do Zlib

Bibliotecas instaladas: libz.so

Descrições Curtas

libz Contém funções de compressão e descompressão usadas por alguns aplicativos

8.7. Bzip2-1.0.8

O pacote Bzip2 contém aplicativos para comprimir e descomprimir arquivos. Comprimir arquivos de texto com **bzip2** gera uma muito melhor percentagem de compressão que com o tradicional **gzip**.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 7,3 MB

8.7.1. Instalação do Bzip2

Aplique um remendo que instalará a documentação para esse pacote:

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

O seguinte comando garante que a instalação de links simbólicos sejam relativos:

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

Garanta que as páginas de manual sejam instaladas no local correto:

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

Prepare o Bzip2 para compilação com:

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

O significado do parâmetro do **make**:

```
-f Makefile-libbz2_so
```

Isso causará Bzip2 ser construído usando um arquivo **Makefile** diferente, nesse caso o arquivo **Makefile-libbz2_so**, o qual cria uma biblioteca dinâmica **libbz2.so** e vincula os utilitários do Bzip2 contra ela.

Compile e teste o pacote:

```
make
```

Instale os aplicativos:

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

Instale a biblioteca compartilhada:

```
cp -av libbz2.so.* /usr/lib
ln -sfv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

O nome da biblioteca compartilhada não é padronizado e varia entre distribuições. A instrução acima instalou **libbz2.so.1.0**, mas alguns aplicativos, por exemplo Kbd, esperam um nome diferente, **libbz2.so.1**, que outras distribuições estão usando. Crie um linque simbólico de compatibilidade para eles:

```
ln -sfv libbz2.so.1.0.8 /usr/lib/libbz2.so.1
```



Nota

A abordagem de linque simbólico só é válida aqui porque a diferença no nome da biblioteca é um resultado de diferentes visões estéticas dos(as) mantenedores(as) de distribuição, não de incompatibilidades reais de ABI. Em geral, uma diferença no nome da biblioteca provavelmente indica uma incompatibilidade de ABI e seria muito provavelmente inválido “ocultar” a diferença por meio de um linque simbólico. Leiam-se Seção 8.2.1, “Problemas de Atualização” para detalhes acerca de nomes de bibliotecas.

Instale o binário compartilhado **bzip2** no diretório `/usr/bin` e substitua duas cópias do **bzip2** por links simbólicos:

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
    ln -sfv bzip2 $i
done
```

Remova uma biblioteca estática inútil:

```
rm -fv /usr/lib/libbz2.a
```

8.7.2. Conteúdo do Bzip2

Programas instalados: bunzip2 (link para bzip2), bzcat (link para bzip2), bzcmp (link para bzdif), bzdif, bzegrep (link para bzgrep), bzfgrep (link para bzgrep), bzgrep, bzip2, bzip2recover, bzless (link para bzmre) e bzmre

Bibliotecas instaladas: libbz2.so

Diretório instalado: /usr/share/doc/bzip2-1.0.8

Descrições Curtas

bunzip2	Descomprime arquivos compactados com bzip
bzcat	Descomprime para a saída padrão
bzcmp	Executa cmp em arquivos compactados com bzip
bzdif	Executa dif em arquivos compactados com bzip
bzegrep	Executa egrep em arquivos compactados com bzip
bzfgrep	Executa fgrep em arquivos compactados com bzip
bzgrep	Executa grep em arquivos compactados com bzip
bzip2	Comprime arquivos usando o algoritmo de compressão de texto de classificação de blocos Burrows-Wheeler com codificação Huffman; a taxa de compressão é melhor que aquela obtida por compressores mais convencionais usando algoritmos “Lempel-Ziv”, como o gzip
bzip2recover	Tenta recuperar dados a partir de arquivos danificados comprimidos com bzip
bzless	Executa less em arquivos compactados com bzip
bzmre	Executa more em arquivos compactados com bzip
libbz2	A biblioteca que implementa compressão de dados de classificação de blocos sem perdas, usando o algoritmo Burrows-Wheeler

8.8. Xz-5.8.3

O pacote Xz contém aplicativos para comprimir e descomprimir arquivos. Ele fornece recursos para os formatos de compressão lzma e o mais novo xz. Comprimir arquivos de texto com **xz** gera uma melhor percentagem de compressão que com os tradicionais comandos **gzip** ou **bzip2**.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 25 MB

8.8.1. Instalação do Xz

Prepare Xz para compilação com:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.8.3
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.8.2. Conteúdo do Xz

Programas instalados:	lzcat (link para xz), lzcmp (link para xzdiff), lzdiff (link para xzdiff), lzegrep (link para xzgrep), lzfgrep (link para xzgrep), lzgrep (link para xzgrep), lzless (link para xzless), lzma (link para xz), lzmadec, lzmainfo, lzmore (link para xzmore), unlzma (link para xz), unxz (link para xz), xz, xzcat (link para xz), xzcmp (link para xzdiff), xzdec, xzdiff, xzegrep (link para xzgrep), xzfgrep (link para xzgrep), xzgrep, xzless e xzmore
Bibliotecas instaladas:	liblzma.so
Diretórios instalados:	/usr/include/lzma e /usr/share/doc/xz-5.8.3

Descrições Curtas

lzcat	Descomprime para a saída padrão
lzcmp	Executa cmp em arquivos comprimidos LZMA
lzdiff	Executa diff em arquivos comprimidos LZMA
lzegrep	Executa egrep em arquivos comprimidos LZMA
lzfgrep	Executa fgrep em arquivos comprimidos LZMA
lzgrep	Executa grep em arquivos comprimidos LZMA
lzless	Executa less em arquivos comprimidos LZMA
lzma	Comprime ou descomprime arquivos usando o formato LZMA
lzmadec	Um decodificador pequeno e rápido para arquivos comprimidos LZMA
lzmainfo	Exibe informação armazenada no cabeçalho de arquivo comprimido com LZMA
lzmore	Executa more em arquivos comprimidos LZMA

unlzma	Descomprime arquivos usando o formato LZMA
unxz	Descomprime arquivos usando o formato XZ
xz	Comprime ou descomprime arquivos usando o formato XZ
xzcat	Descomprime para a saída padrão
xzcmp	Executa cmp em arquivos comprimidos XZ
xzdec	Um decodificador pequeno e rápido para arquivos comprimidos XZ
xzdiff	Executa diff em arquivos comprimidos XZ
xzegrep	Executa egrep em arquivos comprimidos XZ
xzfgrep	Executa fgrep em arquivos comprimidos XZ
xzgrep	Executa grep em arquivos comprimidos XZ
xzless	Executa less em arquivos comprimidos XZ
xzmore	Executa more em arquivos comprimidos XZ
liblzma	A biblioteca que implementa compressão de dados de classificação de blocos, sem perdas, usando o algoritmo de cadeia Lempel-Ziv-Markov

8.9. Lz4-1.10.0

Lz4 é um algoritmo de compressão sem perdas, fornecendo velocidade de compressão superior a 500 MB/s por elemento de processamento. Ele tem um decodificador extremamente rápido, com velocidade de vários GB/s por elemento de processamento. Lz4 pode trabalhar com Zstandard para permitir que ambos os algoritmos comprimam dados mais rapidamente.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 4,2 MB

8.9.1. Instalação do Lz4

Compile o pacote:

```
make BUILD_STATIC=no PREFIX=/usr
```

Para testar os resultados, emita:

```
make -j1 check
```

Instale o pacote:

```
make BUILD_STATIC=no PREFIX=/usr install
```

8.9.2. Conteúdo do Lz4

Programas instalados: lz4, lz4c (link para lz4), lz4cat (link para lz4) e unlz4 (link para lz4)
Biblioteca instalada: liblz4.so

Descrições Curtas

lz4	Comprime ou descomprime arquivos usando o formato LZ4
lz4c	Comprime arquivos usando o formato LZ4
lz4cat	Lista o conteúdo de um arquivo comprimido usando o formato LZ4
unlz4	Descomprime arquivos usando o formato LZ4
liblz4	A biblioteca que implementa compressão de dados sem perdas, usando o algoritmo LZ4

8.10. Zstd-1.5.7

Zstandard é um algoritmo de compressão em tempo real, fornecendo taxas altas de compressão. Ele oferece um intervalo muito amplo de combinações de compressão/velocidade, enquanto é apoiado por um decodificador muito rápido.

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 88 MB

8.10.1. Instalação do Zstd

Compile o pacote:

```
make prefix=/usr
```



Nota

Na saída gerada do teste existem vários lugares que indicam 'failed'. Essas são esperadas e apenas 'FAIL' é uma atual falha de teste. Não deveriam existir falhas de teste.

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make prefix=/usr install
```

Remova a biblioteca estática:

```
rm -v /usr/lib/libzstd.a
```

8.10.2. Conteúdo do Zstd

Programas instalados: zstd, zstdcat (link para zstd), zstdgrep, zstdless, zstdmt (link para zstd) e unzstd (link para zstd)

Biblioteca instalada: libzstd.so

Descrições Curtas

zstd Comprime ou descomprime arquivos usando o formato ZSTD

zstdgrep Executa **grep** em arquivos comprimidos ZSTD

zstdless Executa **less** em arquivos comprimidos ZSTD

libzstd A biblioteca que implementa compressão de dados sem perdas, usando o algoritmo ZSTD

8.11. File-5.48

O pacote File contém um utilitário para determinar o tipo de um dado arquivo ou arquivos.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 21 MB

8.11.1. Instalação do File

Prepare o File para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.11.2. Conteúdo do File

Programas instalados: file

Biblioteca instalada: libmagic.so

Descrições Curtas

file	Tenta classificar cada arquivo dado; ele faz isso realizando vários testes—testes de sistema de arquivos, testes de números mágicos e testes de idioma
libmagic	Contém rotinas para reconhecimento de números mágicos, usadas pelo aplicativo file

8.12. Readline-8.3

O pacote Readline é um conjunto de bibliotecas que oferecem recursos de edição de linha de comando e de histórico.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 17 MB

8.12.1. Instalação do Readline

Reinstalar Readline causará as bibliotecas antigas serem movidas para <nomebiblioteca>.old. Ao tempo em que isso normalmente não seja um problema, em alguns casos isso pode deflagrar um defeito de vinculação no **ldconfig**. Isso pode ser evitado emitindo-se os seguintes dois seds:

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

Evite caminhos de pesquisa de biblioteca de codificação rígida (rpath) dentro das bibliotecas compartilhadas. Esse pacote não precisa do rpath para uma instalação no local padrão, e o rpath às vezes pode causar efeitos indesejados ou até mesmo problemas de segurança:

```
sed -i 's/-Wl,-rpath,[^ ]*//' support/shobj-conf
```

Corrija um problema identificado pelo fluxo de desenvolvimento especificamente para essa versão da readline:

```
sed -e '270a\
    else\
        chars_avail = 1;' \
-e '288i\    result = -1;' \
-i.orig input.c
```

Prepare Readline para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --with-curses \
            --docdir=/usr/share/doc/readline-8.3
```

O significado da nova opção do configure:

--with-curses

Essa opção diz ao Readline que ela pode encontrar as funções da biblioteca termcap na biblioteca curses, não uma biblioteca termcap separada. Isso gerará o arquivo `readline.pc` correto.

Compile o pacote:

```
make SHLIB_LIBS="-lncursesw"
```

O significado da opção do make:

SHLIB_LIBS="-lncursesw"

Essa opção força o Readline a vincular-se à biblioteca `libncursesw`. Para detalhes, veja-se a seção “Bibliotecas Compartilhadas” no arquivo `README` do pacote.

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

Se desejado, instale a documentação:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.3
```

8.12.2. Conteúdo do Readline

Bibliotecas instaladas: libhistory.so e libreadline.so
Diretórios instalados: /usr/include/readline e /usr/share/doc/readline-8.3

Descrições Curtas

libhistory	Fornece uma consistente interface de usante para rechamar linhas do histórico
libreadline	Fornece um conjunto de comandos para manipular texto digitado em uma sessão interativa de um aplicativo

8.13. Pcre2-10.47

O pacote pcre2 contém uma nova geração das bibliotecas Perl Compatible Regular Expression.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 28 MB

8.13.1. Instalação do Pcre2

Prepare pcre2 para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/pcre2-10.47 \
            --enable-unicode \
            --enable-jit \
            --enable-pcre2-16 \
            --enable-pcre2-32 \
            --enable-pcre2grep-libz \
            --enable-pcre2grep-libbz2 \
            --enable-pcre2test-libreadline \
            --disable-static
```

O significado das novas opções de configurar:

--enable-unicode

Essa opção habilita o suporte Unicode e inclui as funções para manusear sequências de caracteres UTF-8/16/32 na biblioteca.

--enable-jit

Essa opção habilita compilação just-in-time, o que pode acelerar bastante a correspondência de padrões.

--enable-pcre2-16

Essa opção habilita suporte a caracteres de 16 bits.

--enable-pcre2-32

Essa opção habilita suporte a caracteres de 32 bits.

--enable-pcre2grep-libz

Essa opção adiciona suporte para leitura de arquivos comprimidos .gz no pcre2grep.

--enable-pcre2grep-libbz2

Essa opção adiciona suporte para leitura de arquivos comprimidos .bz2 ao pcre2grep.

--enable-pcre2test-libreadline

Essa opção adiciona recursos de edição de linha e de histórico ao programa pcre2test.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.13.2. Conteúdo do Pcre2

Programas instalados: pcre2grep e pcre2test

Biblioteca instalada: libpcre2-8.so, libpcre2-16.so, libpcre2-32.so e libpcre2-posix.so

Descrições Curtas

pcre2grep	é uma versão do grep que entende expressões regulares compatíveis com Perl
pcre2test	consegue testar uma expressão regular compatível com Perl

8.14. M4-1.4.21

O pacote M4 contém um processador de macro.

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 62 MB

8.14.1. Instalação do M4

Prepare o M4 para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.14.2. Conteúdo de M4

Aplicativo instalado: m4

Descrições Curtas

m4 Copia os arquivos dados enquanto expande as macros que eles contém. Essas macros são ou nativas ou definidas pelo(a) usante e podem receber qualquer número de argumentos. Além de executar expansão de macro, **m4** tem funções nativas para incluir arquivos nomeados, executar comandos Unix, realizar aritmética de inteiros, manipular texto, recursão, entre outras. O programa **m4** pode ser usado ou como uma estrutura de interação direta com o(a) usante para um compilador ou como um processador de macro independente

8.15. Bc-7.0.3

O pacote Bc contém uma linguagem de processamento numérica de precisão arbitrária.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 7,8 MB

8.15.1. Instalação do Bc

Prepare Bc para compilação:

```
CC='gcc -std=c99' ./configure --prefix=/usr -G -O3 -r
```

O significado das opções do configure:

CC='gcc -std=c99'

Esse parâmetro especifica o compilador padrão C a se usar.

-G

Omite partes da suíte de teste que não funcionariam até que o aplicativo bc tenha sido instalado.

-O3

Especifica a otimização a usar.

-r

Habilita o uso de Readline para melhorar o recurso de edição de linha do bc.

Compile o pacote:

```
make
```

Para testar bc, execute:

```
make test
```

Instale o pacote:

```
make install
```

8.15.2. Conteúdo do Bc

Programas instalados: bc e dc

Descrições Curtas

bc Uma calculadora de linha de comando

dc Uma calculadora de linha de comando de polonesa - reversa

8.16. Flex-2.6.4

O pacote Flex contém um utilitário para gerar programas que reconhecem padrões em texto.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 33 MB

8.16.1. Instalação do Flex

Prepare Flex para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/flex-2.6.4
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

Uns poucos aplicativos não sabem acerca do **flex** ainda e tentam executar o predecessor dele, **lex**. Para suportar esses aplicativos, crie um link simbólico chamado `lex` que executa o `flex` no modo de emulação **lex**, e também crie a página de manual do **lex** como um link simbólico:

```
ln -sv flex /usr/bin/lex
ln -sv flex.1 /usr/share/man/man1/lex.1
```

8.16.2. Conteúdo do Flex

Programas instalados: flex, flex++ (link para flex) e lex (link para flex)

Bibliotecas instaladas: libfl.so

Diretório instalado: /usr/share/doc/flex-2.6.4

Descrições Curtas

flex	Uma ferramenta para gerar aplicativos que reconhecem padrões em texto; ela permite, para a versatilidade, especificar as regras para encontrar padrões, erradicando a necessidade de desenvolver um aplicativo especializado
flex++	Uma extensão do flex, é usada para gerar código e classes C++. É um link simbólico para flex
lex	Um link simbólico que executa o flex no modo de emulação lex
<code>libfl</code>	A biblioteca flex

8.17. Tcl-8.6.18

O pacote Tcl contém a Tool Command Language, uma linguagem de script robusta de propósito geral. O pacote Expect é escrito em Tcl (pronunciada "tickle").

Tempo aproximado de construção: 2,9 UPC

Espaço em disco exigido: 92 MB

8.17.1. Instalação do Tcl

Esse pacote e os próximos dois (Expect e DejaGNU) são instalados para suportar a execução das suítes de teste para Binutils, GCC e outros pacotes. Instalar três pacotes para propósitos de teste possivelmente pareça excessivo, mas é muito assegurador, se não essencial, saber que as ferramentas mais importantes estão funcionando adequadamente.

Prepare Tcl para compilação:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr \
            --mandir=/usr/share/man \
            --disable-rpath
```

O significado dos novos parâmetros de configuração:

--disable-rpath

Esse parâmetro evita caminhos de pesquisa de biblioteca de codificação rígida (rpath) dentro dos arquivos binários executáveis e das bibliotecas compartilhadas. Esse pacote não precisa do rpath para uma instalação no local padrão, e o rpath às vezes pode causar efeitos indesejados ou até mesmo problemas de segurança.

Construa o pacote:

```
make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|" \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.13|/usr/lib/tdbc1.1.13|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.13/generic|/usr/include|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.13/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.13|/usr/include|" \
    -i pkgs/tdbc1.1.13/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.3.7|/usr/lib/itcl4.3.7|" \
    -e "s|$SRCDIR/pkgs/itcl4.3.7/generic|/usr/include|" \
    -e "s|$SRCDIR/pkgs/itcl4.3.7|/usr/include|" \
    -i pkgs/itcl4.3.7/itclConfig.sh

unset SRCDIR
```

As várias instruções “sed” depois do comando “make” removem referências ao diretório de construção dos arquivos de configuração e as substituem pelo diretório de instalação. Isso não é obrigatório para o restante do LFS, porém possivelmente seja necessário se um pacote construído posteriormente usar a Tcl.

Para testar os resultados, emita:

```
LC_ALL=C.UTF-8 make test
```

Instale o pacote:

```
make install
chmod 644 /usr/lib/libtclstub8.6.a
```

Torne as bibliotecas instaladas graváveis, de modo que símbolos de depuração possam ser removidos posteriormente:

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

Instale os cabeçalhos da Tcl. O próximo pacote, Expect, exige elas.

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

Agora faça um link simbólico necessário:

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Renomeie uma página de manual que conflita com uma página de manual do Perl:

```
mv -v /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

Opcionalmente, instale a documentação emitindo os seguintes comandos:

```
cd ..
tar -xf ../tcl8.6.18-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.18
cp -v -r ./html/* /usr/share/doc/tcl-8.6.18
```

8.17.2. Conteúdo do Tcl

Programas instalados: tclsh (link para tclsh8.6) e tclsh8.6
Biblioteca instalada: libtcl8.6.so e libtclstub8.6.a

Descrições Curtas

tclsh8.6	O shell de comando da Tcl
tclsh	Um link para tclsh8.6
libtcl8.6.so	A biblioteca Tcl
libtclstub8.6.a	A biblioteca Stub da Tcl

8.18. Expect-5.45.4

O pacote Expect contém ferramentas para automatizar, via diálogos com script, aplicativos interativos, tais como o **telnet**, **ftp**, **passwd**, **fsck**, **rlogin** e **tip**. Expect também é útil para testar esses mesmos aplicativos, bem como para facilitar todos os tipos de tarefas que são proibitivamente difíceis com qualquer outra coisa. A estrutura subjacente da DejaGnu é escrita em Expect.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 3,9 MB

8.18.1. Instalação do Expect

Expect precisa de PTYs para funcionar. Verifique se os PTYs estão funcionando corretamente dentro do ambiente chroot realizando um teste simples:

```
python3 -c 'from pty import spawn; spawn(["echo", "ok"])'
```

Esse comando deveria gerar ok. Se, em vez disso, a saída gerada incluir `osError: out of pty devices`, então o ambiente não está configurado para operação adequada de PTY. Você precisa sair do ambiente chroot, ler Seção 7.3, “Preparando Sistemas de Arquivos Virtuais do Núcleo” novamente e garantir que o sistema de arquivos `devpts` (e outros sistemas de arquivos virtuais do núcleo) foi montado corretamente. Em seguida, entre novamente no ambiente chroot seguindo Seção 7.4, “Entrando no Ambiente Chroot”. Esse problema precisa ser resolvido antes de continuar ou as suítes de teste que exigem o Expect (por exemplo, as suítes de teste do Bash, Binutils, GCC, GDBM e, com certeza, a próprio Expect) falharão catastroficamente e outras falhas sutis possivelmente também aconteçam.

Agora, faça algumas mudanças para permitir o pacote com gcc-15.1 ou posterior:

```
patch -Np1 -i ../expect-5.45.4-gcc15-1.patch
```

Prepare Expect para compilação:

```
./configure --prefix=/usr \
            --with-tcl=/usr/lib \
            --enable-shared \
            --disable-rpath \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

O significado das opções do configure:

`--with-tcl=/usr/lib`

Esse parâmetro é necessário para dizer ao **configure** onde o script **tclConfig.sh** está localizado.

`--with-tclinclude=/usr/include`

Isso explicitamente diz a Expect onde encontrar os cabeçalhos internos da Tcl.

Construa o pacote:

```
make
```

Para testar os resultados, emita:

```
make test
```

Instale o pacote:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.18.2. Conteúdo do Expect

Aplicativo instalado: expect

Biblioteca instalada: libexpect5.45.4.so

Descrições Curtas

expect	Comunica-se com outros aplicativos interativos de acordo com um script
<code>libexpect-5.45.4.so</code>	Contém funções que permitem a Expect ser usado como uma extensão da Tcl ou ser usado diretamente a partir de C ou C++ (sem a Tcl)

8.19. DejaGNU-1.6.3

O pacote DejaGnu contém uma estrutura subjacente para executar suítes de teste em ferramentas GNU. Ele é escrito em **expect**, a qual usa ela própria a Tcl (Tool Command Language).

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 7,4 MB

8.19.1. Instalação do DejaGNU

A(O) desenvolvedora(r) recomenda construir DejaGNU em um diretório dedicado à construção:

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

Prepare DejaGNU para compilação:

```
../configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext      -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
install -v -dm755 /usr/share/doc/dejagnu-1.6.3
install -v -m644 doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

8.19.2. Conteúdo do DejaGNU

Aplicativo instalado: dejagnu e runtest

Descrições Curtas

dejagnu Iniciador de comando auxiliar DejaGNU

runtest Um script encapsulador que localiza o shell **expect** adequado e, em seguida, executa o DejaGNU

8.20. Ninja-1.13.2

Ninja é um sistema de construção pequeno com um foco em velocidade.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 43 MB

8.20.1. Instalação do Ninja

Quando executado, **ninja** normalmente utiliza o maior número possível de processos em paralelo. Por padrão, esse é o número de núcleos no sistema, mais dois. Isso possivelmente superaqueça a CPU ou faça o sistema ficar sem memória. Quando **ninja** é invocado a partir da linha de comando, passar o parâmetro `-jN` limitará o número de processos paralelos. Alguns pacotes embutem a execução do **ninja** e não passam o parâmetro `-j` para ele.

Usar o procedimento *opcional* abaixo permite que um(a) usante limite o número de processos paralelos via uma variável de ambiente, `NINJAJOBS`. **Por exemplo**, configurar:

```
export NINJAJOBS=4
```

limitará **ninja** a quatro processos paralelos.

Se desejado, [então] faça o **ninja** reconhecer a variável de ambiente `NINJAJOBS` executando o editor de fluxo:

```
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
```

Construa Ninja com:

```
python3 configure.py --bootstrap --verbose
```

O significado da opção de construção:

`--bootstrap`

Esse parâmetro força Ninja a reconstruir ele próprio para o sistema atual.

`--verbose`

Esse parâmetro faz com que **configure.py** mostre o progresso da construção do Ninja.

Os testes de pacote não podem executar no ambiente chroot. Eles exigem *cmake*. Mas a função básica desse pacote já foi testada pela reconstrução dele mesmo (com a opção `--bootstrap`) de qualquer forma.

Instale o pacote:

```
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
```

8.20.2. Conteúdo do Ninja

Programas instalados: `ninja`

Descrições Curtas

ninja é o sistema de construção Ninja

8.21. Pkgconf-3.0.5

O pacote pkgconf é um sucessor do pkg-config e contém uma ferramenta para passar o caminho "include" e(ou) caminhos de biblioteca para construir ferramentas durante as fases "configure" e "make" de instalações de pacotes.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 47 MB

8.21.1. Instalação do Pkgconf

Primeiro, contorne uma dependência circular do meson:

```
tar -xf ../meson-1.12.0.tar.gz
```

Prepare Pkgconf para compilação:

```
mkdir build
cd build

python3 ../meson-1.12.0/meson.py setup --prefix=/usr --buildtype=release ..
```

Compile o pacote:

```
ninja
```

Para testar os resultados, emita:

```
ninja test
```

Instale o pacote:

```
ninja install
mv /usr/share/doc/pkgconf{,-3.0.5}
```

Para manter a compatibilidade com o Pkg-config original, crie dois links simbólicos:

```
ln -sv pkgconf /usr/bin/pkg-config
ln -sv pkgconf.1 /usr/share/man/man1/pkg-config.1
```

8.21.2. Conteúdo do Pkgconf

Programas instalados: pkgconf, pkg-config (link para pkgconf) e bomtool

Biblioteca instalada: libpkgconf.so

Diretório instalado: /usr/share/doc/pkgconf-3.0.5

Descrições Curtas

pkgconf Retorna metainformações para a biblioteca ou pacote especificada

bomtool Gera uma Lista de Materiais do Software a partir de arquivos ".pc" do "pkg-config"

libpkgconf.so Contém a maior parte da funcionalidade do "pkgconf", enquanto permite que outras ferramentas, como "IDEs" e compiladores, usem a estrutura essencial de suporte dele

8.22. Binutils-2.47

O pacote Binutils contém um vinculador, um montador e outras ferramentas para manusear arquivos objeto.

Tempo aproximado de construção: 1,7 UPC

Espaço em disco exigido: 817 MB

8.22.1. Instalação do Binutils

A documentação do Binutils recomenda construir o Binutils em um diretório dedicado à construção:

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

Prepare o Binutils para compilação:

```
../configure --prefix=/usr \
             --sysconfdir=/etc \
             --enable-lld=default \
             --enable-plugins \
             --enable-shared \
             --disable-werror \
             --enable-64-bit-bfd \
             --enable-new-dtags \
             --with-system-zlib \
             --with-lib-path=/usr/lib \
             --enable-default-hash-style=gnu
```

O significado dos novos parâmetros de configuração:

--enable-lld=default

Constrói o vinculador bfd original e o instala como ambos ld (o vinculador padrão) e ld.bfd.

--enable-plugins

Habilite suporte de plugin para o lincador.

--with-system-zlib

Usa a biblioteca zlib instalada em vez de construir a versão incluída.

--with-lib-path=/usr/lib

Especifique o caminho para o lincador (**ld**) procurar. Por padrão, ele procura em vários diretórios que não existem no LFS, além de `/usr/lib`, especialmente o diretório `/usr/lib64`, que nós deliberadamente evitamos. No caso onde `/usr/lib64` tenha sido criado e povoado com algumas bibliotecas por engano, fazer com que **ld** não procure nesse caminho pode realçar o problema mais cedo com uma falha para localizar essas bibliotecas em tempo de construção em vez de em tempo de execução.

Compile o pacote:

```
make tooldir=/usr
```

O significado do parâmetro do make:

tooldir=/usr

Normalmente, o tooldir (o diretório onde os executáveis ultimamente estarão localizados) é configurado para `$(exec_prefix)/$(target_alias)`. Por exemplo, máquinas x86_64 expandiriam isso para `/usr/x86_64-linux-gnu`. Por causa que este é um sistema personalizado, esse diretório alvo específico em `/usr` não é exigido. `$(exec_prefix)/$(target_alias)` seria usado se o sistema fosse usado para compilar cruzadamente (por exemplo, compilar um pacote em uma máquina Intel que gera código que pode ser executado em máquinas PowerPC).



Importante

A suíte de teste para Binutils nesta seção é considerada crítica. Não a pule sob quaisquer circunstâncias.

Teste os resultados:

```
make -k check
```

Para uma lista de testes falhos, execute:

```
grep '^FAIL:' $(find -name '*.log')
```

One teste relacionado a gprofng é conhecido por falhar.

Instale o pacote:

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

Remova bibliotecas estáticas inúteis e outros arquivos:

```
rm -rfv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a \
/usr/share/doc/gprofng/
```

8.22.2. Conteúdo do Binutils

Programas instalados:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings e strip
Bibliotecas instaladas:	libbfd.so, libctf.so, libctf-nobfd.so, libgprofng.so, libopcodes.so e libsframe.so
Diretório instalado:	/usr/lib/ldscripts

Descrições Curtas

addr2line	Traduz endereços de aplicativos para nomes de arquivo e números de linha; dado um endereço e o nome de um executável, ele usa a informação de depuração no executável para determinar qual arquivo fonte e número de linha estão associados com o endereço
ar	Cria, modifica e extrai a partir de arquivamentos
as	Um montador que monta a saída gerada do gcc para dentro de arquivos objeto
c++filt	Usado pelo vinculador para desmembrar símbolos C++ e Java e para impedir que funções sobrecarregadas entrem em conflito
dwp	O utilitário de empacotamento DWARF
elfedit	Atualiza o cabeçalho ELF de arquivos ELF
gprof	Exibe dados do perfil de gráfico de chamada
gprofng	Coleta e analisa dados de desempenho
ld	Um vinculador que combina um número de objetos e arquivos de arquivamento em um arquivo, realocando os dados deles e vinculando referências de símbolos
ld.bfd	Um link rígido para o ld
nm	Lista os símbolos ocorrentes em um dado arquivo objeto
objcopy	Traduz um tipo de arquivo objeto em outro
objdump	Exibe informação a respeito do dado arquivo objeto, com opções controlando a informação particular a exibir; a informação mostrada é útil para programadores(as) que estão trabalhando nas ferramentas de compilação

ranlib	Gera um índice do conteúdo de um arquivamento e o armazena no arquivamento; o índice lista todos os símbolos definidos pelos membros do arquivamento que são arquivos objeto realocáveis
readelf	Exibe informação a respeito de binários do tipo ELF
size	Lista os tamanhos de seção e o tamanho total para os arquivos objeto dados
strings	Exibe, para cada arquivo dado, as sequências de caracteres imprimíveis que são, no mínimo, do comprimento especificado (padronizado para quatro); para arquivos objeto, ele imprime, por padrão, somente as sequências de caracteres a partir das seções de inicialização e carregamento enquanto que para outros tipos de arquivos, ele escaneia o arquivo inteiro
strip	Descarta símbolos originários de arquivos objeto
<code>libbfd</code>	A biblioteca de Descritor de Arquivo Binário
<code>libctf</code>	A biblioteca de suporte de depuração Compat ANSI-C Type Format
<code>libctf-nobfd</code>	Uma variante da <code>libctf</code> que não usa funcionalidade da <code>libbfd</code>
<code>libgprofng</code>	Uma biblioteca contendo a maioria das rotinas usadas pelo gprofng
<code>libopcodes</code>	Uma biblioteca para lidar com opcodes—as versões de “texto legível” de instruções para o processador; é usada para construir utilitários como o objdump
<code>libsframe</code>	Uma biblioteca para suportar retrocesso em linha usando um desbobinador simples

8.23. GMP-6.3.0

O pacote GMP contém bibliotecas matemáticas. Essas tem funções úteis para aritmética de precisão arbitrária.

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 54 MB

8.23.1. Instalação do GMP



Nota

Se você estiver construindo para x86 de 32 bits, mas tem uma CPU que seja capaz de executar código de 64 bits e você especificou `CFLAGS` no ambiente, [então] o script `configure` tentará configurar para 64 bits e falhará. Impeça isso invocando o comando do `configure` abaixo com

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



Nota

As configurações padrão do GMP produzem bibliotecas otimizadas para o processador anfitrião. Se bibliotecas adequadas para processadores menos capazes que a CPU do anfitrião forem desejadas, bibliotecas genéricas podem ser criadas pospondo-se a opção `--host=none-linux-gnu` ao comando `configure`.

Primeiro, faça um ajustamento para compatibilidade com gcc-15 e posteriores:

```
sed -i '/long long t1;/,+1s/()/(...)/' configure
```

Prepare GMP para compilação:

```
./configure --prefix=/usr \
            --enable-cxx \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.3.0
```

O significado das novas opções de configurar:

`--enable-cxx`

Esse parâmetro habilita suporte a C++

`--docdir=/usr/share/doc/gmp-6.3.0`

Essa variável especifica o lugar correto para a documentação.

Compile o pacote e gere a documentação HTML:

```
make
make html
```



Importante

A suíte de teste para o GMP nesta seção é considerada crítica. Não a pule sob quaisquer circunstâncias.

Teste os resultados:

```
make check
```



Cuidado

O código em "gmp" é altamente otimizado para o processador onde ele seja construído. Ocasionalmente, o código que detecta o processador identifica erroneamente os recursos do sistema e existirão erros nos testes ou em outros aplicativos usando as bibliotecas "gmp" com a mensagem `Illegal instruction`. Nesse caso, o "gmp" deveria ser reconfigurado com a opção `--host=none-linux-gnu` e reconstruído.

Certifique-se de que pelo menos 199 testes na suíte de teste passaram. Verifique os resultados emitindo o seguinte comando:

```
cat $(find -name '*.log') | grep -c ^PASS
```

Instale o pacote e a documentação dele:

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

8.23.2. Conteúdo do GMP

Bibliotecas instaladas: libgmp.so e libgmpxx.so
Diretório instalado: /usr/share/doc/gmp-6.3.0

Descrições Curtas

libgmp	Contém funções matemáticas de precisão
libgmpxx	Contém funções matemáticas de precisão C++

8.24. MPFR-4.2.2

O pacote MPFR contém funções para matemática de precisão múltipla.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 44 MB

8.24.1. Instalação do MPFR

Prepare MPFR para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.2
```

Compile o pacote e gere a documentação HTML:

```
make
make html
```



Importante

A suíte de teste para o MPFR nesta seção é considerada crítica. Não a pule sob quaisquer circunstâncias.

Teste os resultados e certifique-se de que todos os 198 testes passaram:

```
make check
```

Instale o pacote e a documentação dele:

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

8.24.2. Conteúdo do MPFR

Bibliotecas instaladas: libmpfr.so

Diretório instalado: /usr/share/doc/mpfr-4.2.2

Descrições Curtas

libmpfr Contém funções matemáticas de precisão múltipla

8.25. MPC-1.4.1

O pacote MPC contém uma biblioteca para a aritmética de números complexos com precisão arbitrariamente alta e arredondamento correto do resultado.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 23 MB

8.25.1. Instalação do MPC

Prepare MPC para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.4.1
```

Compile o pacote e gere a documentação HTML:

```
make
make html
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote e a documentação dele:

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

8.25.2. Conteúdo do MPC

Bibliotecas instaladas: libmpc.so

Diretório instalado: /usr/share/doc/mpc-1.4.1

Descrições Curtas

`libmpc` Contém funções matemáticas complexas

8.26. Attr-2.6.0

O pacote Attr contém utilitários para administrar os atributos estendidos dos objetos do sistema de arquivos.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 4,5 MB

8.26.1. Instalação do Attr

Prepare Attr para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.6.0
```

Compile o pacote:

```
make
```

Os testes precisam ser executados sobre um sistema de arquivos que suporte atributos estendidos, tais como os sistemas de arquivos ext2, ext3 ou ext4. Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.26.2. Conteúdo do Attr

Programas instalados: attr, getfattr e setfattr
Biblioteca instalada: libattr.so
Diretórios instalados: /usr/include/attr e /usr/share/doc/attr-2.6.0

Descrições Curtas

attr	Estende atributos sobre objetos dos sistemas de arquivos
getfattr	Obtém os atributos estendidos dos objetos do sistema de arquivos
setfattr	Configura os atributos estendidos dos objetos do sistema de arquivos
libattr	Contém as funções de biblioteca para manipular atributos estendidos

8.27. Acl-2.4.0

O pacote Acl contém utilitários para administrar Listas de Controle de Acesso, as quais são usadas para definir direitos de acesso discricionários refinados para arquivos e diretórios.

Tempo aproximado de menos que 0,1 UPC

construção:

Espaço em disco exigido: 7,1 MB

8.27.1. Instalação do Acl

Prepare Acl para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/acl-2.4.0
```

Compile o pacote:

```
make
```

Os testes de Acl precisam ser executados em um sistema de arquivos que suporte controles de acesso. Para testar os resultados, emita:

```
make check
```

Um teste nominado `test/cp.run` é conhecido por falhar porque Coreutils ainda não foi construído com o suporte de Acl.

Instale o pacote:

```
make install
```

8.27.2. Conteúdo do Acl

Programas instalados: chacl, getfacl e setfacl

Biblioteca instalada: libacl.so

Diretórios instalados: /usr/include/acl e /usr/share/doc/acl-2.4.0

Descrições Curtas

chacl Muda a lista de controle de acesso de um arquivo ou diretório

getfacl Obtém listas de controle de acesso do arquivo

setfacl Configura listas de controle de acesso do arquivo

libacl Contém as funções de biblioteca para manipular Listas de Controle de Acesso

8.28. Libcap-2.78

O pacote Libcap implementa a interface do espaço de usuário para os recursos POSIX 1003.1 e disponíveis em núcleos Linux. Esses recursos particionam o todo poderoso privilégio de root em um conjunto de privilégios distintos.

Tempo aproximado de menos que 0,1 UPC

construção:

Espaço em disco exigido: 3,1 MB

8.28.1. Instalação do Libcap



Nota

Se atualizar esse pacote em um sistema existente e o compilador Go estiver instalado, evite um erro de construção usando **export GOLANG=no** antes de executar os comandos abaixo. Certifique-se de desconfigurar GOLANG depois que a instalação estiver completa.

Impedindo bibliotecas estáticas de serem instaladas:

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

Compile o pacote:

```
make prefix=/usr lib=lib
```

O significado da opção do make:

lib=lib

Esse parâmetro configura o diretório de biblioteca para `/usr/lib` em vez de `/usr/lib64` em `x86_64`. Ele não tem efeito em `x86`.

Para testar os resultados, emita:

```
make test
```

Instale o pacote:

```
make prefix=/usr lib=lib install
```

8.28.2. Conteúdo do Libcap

Programas instalados: capsh, getcap, getpcaps e setcap

Biblioteca instalada: libcap.so e libpsx.so

Descrições Curtas

capsh	Um encapsulador de shell para explorar e restringir suporte a recurso
getcap	Examina recursos do arquivo
getpcaps	Exibe os recursos do(s) processo(s) consultado(s)
setcap	Configura recursos do arquivo
libcap	Contém as funções de biblioteca para manipular recursos POSIX 1003.1e
libpsx	Contém funções para suportar semântica POSIX para chamadas de sistema associadas com a biblioteca pthread

8.29. Libxcrypt-4.5.2

O pacote Libxcrypt contém uma biblioteca moderna para hash unidirecional de senhas.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 14 MB

8.29.1. Instalação do Libxcrypt

Primeiro, faça uma correção exigida pela glibc-2.43 e posteriores:

```
sed -i '/strchr/s/const//' lib/crypt-{sm3,gost}-yescrypt.c
```

Prepare Libxcrypt para compilação:

```
./configure --prefix=/usr \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=no \
            --disable-static \
            --disable-failure-tokens
```

O significado das novas opções de configurar:

--enable-hashes=strong,glibc

Constrói algoritmos fortes de resumo recomendados para casos de uso de segurança e os algoritmos de resumo fornecidos pela tradicional `libcrypt` da "Glibc" para compatibilidade.

--enable-obsolete-api=no

Desabilita as funções obsoletas da API. Elas não são necessárias para um sistema moderno Linux construído a partir do fonte.

--disable-failure-tokens

Desabilita o recurso de ficha de falha. É necessário para compatibilidade com as bibliotecas tradicionais de resumo de algumas plataformas, mas um sistema Linux baseado na "Glibc" não precisa dele.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```



Nota

As instruções acima desabilitaram funções obsoletas da API, pois nenhum pacote instalado por compilação a partir dos fontes se vincularia a elas em tempo de execução. No entanto, os únicos aplicativos somente binários conhecidos que se vinculam a essas funções exigem ABI versão 1. Se você precisar ter tais funções devido a algum aplicativo somente binário ou para estar conforme com a "LSB", [então] construa o pacote novamente com os seguintes comandos:

```
make distclean
./configure --prefix=/usr \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=glibc \
            --disable-static \
            --disable-failure-tokens
make
cp -av --remove-destination .libs/libcrypt.so.1* /usr/lib
```

8.29.2. Conteúdo do Libxcrypt

Bibliotecas instaladas: `libcrypt.so`

Descrições Curtas

`libcrypt` Contém funções para resumir senhas

8.30. Shadow-4.20.2

O pacote Shadow contém aplicativos para manusear senhas de uma maneira segura.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 112 MB

8.30.1. Instalação do Shadow



Importante

Se você tiver instalado o Linux-PAM, você deveria seguir *as instruções do BLFS* em vez desta página para construir, reconstruir, atualizar shadow.



Nota

Se você gostaria de impor o uso de senhas fortes, *instale e configure o Linux-PAM* primeiro. Então *instale e configure o shadow com o suporte PAM*. Finalmente *instale o libpwquality e configure o PAM para usá-lo*.

Impeça a instalação de páginas de manual que já foram instaladas no Seção 8.3, “Man-pages-6.18”:

```
find man -name Makefile.in -exec sed -i 's/getsppam\3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\5 / /' {} \;
```

Em vez de usar o método padrão *SHA512*, use o método mais seguro *YESCRIPT* de encriptação de senha. Também é necessário mudar o local obsoleto `/var/spool/mail` para caixas de correio de usante que Shadow usa por padrão para o local `/var/mail` usado atualmente. E, remova `/bin` e `/sbin` do `PATH`, pois eles são simplesmente linques simbólicos para as contrapartes deles em `/usr`.



Atenção

Incluir `/bin` e (ou) `/sbin` na variável `PATH` pode fazer com que alguns pacotes do BLFS falhem para construir, de forma que não faça isso no arquivo `.bashrc` ou em qualquer outro lugar.

```
sed -e 's:#ENCRYPT_METHOD SHA512:ENCRYPT_METHOD YESCRYPT:' \
-e 's:/var/spool/mail:/var/mail:' \
-e '/PATH={s@/sbin:@@;s@/bin:@@}' \
-i etc/login.defs
```

Prepare o Shadow para compilação:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
--disable-static \
--with-{b,yes}crypt \
--without-libbsd \
--disable-logind \
--with-group-name-max-length=32
```

O significado das novas opções de configuração:

touch /usr/bin/passwd

O arquivo `/usr/bin/passwd` precisa existir, pois o local dele é codificado rigidamente em alguns aplicativos; se ele já não existir, [então] o conjunto de comandos sequenciais de instalação o criará no lugar errado.

--with-{b,yes}crypt

O shell expande isso para duas chaves, `--with-bcrypt` e `--with-yescrypt`. Elas permitem que o shadow use os algoritmos Bcrypt e Yescrypt implementados pelo Libxcrypt para resumir senhas. Esses algoritmos são mais seguros (em particular, muito mais resistentes a ataques baseados em GPU) que os algoritmos SHA tradicionais.

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

O nome de usante mais longo permissível é o de trinta e dois (32) caracteres. Torne o comprimento máximo de um nome de grupo o mesmo.

```
--disable-logind
```

Essa opção faz com que o Shadow (especificamente, os programas **login** e **who**) use o arquivo `/run/utmp` em vez do `logind` para rastrear as sessões ativas de login, pois o `logind` ainda não está disponível no sistema LFS incompleto. Mas como nós discutimos em Seção 7.6, “Criando Arquivos Essenciais e Linques Simbólicos”, o formato de arquivo `/run/utmp` estará completamente quebrado depois do ano 2038. Os(As) editores(as) do LFS tentarão resolver o problema antes desse ano.

```
--without-libbsd
```

Não use a função “`readpassphrase`” originária da “`libbsd`” a qual não está no LFS. Use a cópia interna.

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make exec_prefix=/usr install
make -C man install-man
```

8.30.2. Configurando o Shadow

Esse pacote contém utilitários para adicionar, modificar e deletar usantes e grupos; configurar e mudar as senhas deles(as); e realizar outras tarefas administrativas. Para uma explicação completa do que *sombreamento de senha* significa, veja-se o arquivo `doc/HOWTO` dentro da árvore desempacotada do fonte.

Habilite senhas sombreadas:

```
pwconv
```

Habilite senhas sombreadas de grupo também:

```
grpconv
```

A configuração padrão do Shadow para o utilitário **useradd** precisa de alguma explicação. Primeiro, a ação padrão para o utilitário **useradd** é a de criar o(a) usante e um grupo com o mesmo nome que o(a) usante. Por padrão, os números de ID de usante (UID) e de ID de grupo (GID) iniciarão em 1000. Isso significa que, se você não passar parâmetros extras para **useradd**, cada usante será um(a) membro(a) de um grupo único no sistema. Se esse comportamento for indesejável, você precisará passar ou o parâmetro `-g` ou o `-N` para **useradd**, ou, do contrário, mudar a configuração de `USERGROUPS_ENAB` em `/etc/login.defs`. Veja-se *useradd(8)* para mais informação.

Segundo, para mudar os parâmetros padrão, o arquivo `/etc/default/useradd` precisa ser criado e adaptado para se adequar às tuas necessidades particulares. Crie-o com:

```
mkdir -p /etc/default
useradd -D --gid 999
```

Explicações do parâmetro do `/etc/default/useradd`

`GROUP=999`

Esse parâmetro configura o início dos números de grupo usados no arquivo `/etc/group`. O valor específico 999 vem do parâmetro `--gid` acima. Você possivelmente o configure para qualquer valor desejado. Observe que **useradd** nunca reusará um UID ou um GID. Se o número identificado nesse parâmetro estiver usado, ele usará o próximo número disponível. Observe também que, se você não tiver um grupo com um ID igual a esse número em teu sistema, então na primeira vez que você usar o **useradd** sem o parâmetro `-g`, uma mensagem

de erro será gerada—`useradd: unknown GID 999`, ainda que a conta tenha sido criada corretamente. Esse é o motivo pelo qual nós criamos o grupo `users` com esse ID de grupo no Seção 7.6, “Criando Arquivos Essenciais e Linques Simbólicos.”

```
CREATE_MAIL_SPOOL=yes
```

Esse parâmetro faz com que o **useradd** crie um arquivo de caixa de correio para cada novo(a) usante. **useradd** atribuirá a titularidade de propriedade de grupo desse arquivo para o grupo `mail` com permissões `0660`. Se você, em vez disso, não quisesse criar esses arquivos, emita o seguinte comando:

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

Finalmente, crie os arquivos vazios `/etc/subuid` e `/etc/subgid`:

```
touch /etc/sub{u,g}id
```

O conteúdo desses arquivos será atualizado automaticamente por alguns utilitários fornecidos pelo Shadow para alocar IDs de usante e de grupo subordinados. Leia-se as páginas de manual *subuid(5)* e *subgid(5)* para os detalhes acerca dos IDs subordinados.

8.30.3. Configurando a Senha do(a) Root

Escolha uma senha para o(a) usante *root* e configure-a executando:

```
passwd root
```

8.30.4. Conteúdo do Shadow

Programas instalados:	<code>chage</code> , <code>chfn</code> , <code>chgpaswd</code> , <code>chpaswd</code> , <code>chsh</code> , <code>faillog</code> , <code>getsubids</code> , <code>gpaswd</code> , <code>groupadd</code> , <code>groupdel</code> , <code>groupmod</code> , <code>grpck</code> , <code>grpconv</code> , <code>grpunconv</code> , <code>login</code> , <code>newgidmap</code> , <code>newgrp</code> , <code>newuidmap</code> , <code>newusers</code> , <code>nologin</code> , <code>passwd</code> , <code>pwck</code> , <code>pwconv</code> , <code>pwunconv</code> , <code>sg</code> (linque para <code>newgrp</code>), <code>su</code> , <code>useradd</code> , <code>userdel</code> , <code>usermod</code> , <code>vigr</code> (linque para <code>vipw</code>) e <code>vipw</code>
Bibliotecas instaladas:	<code>libsubid.so</code>
Diretórios instalados:	<code>/etc/default</code> e <code>/usr/include/shadow</code>

Descrições Curtas

chage	Usado para mudar o número máximo de dias entre mudanças obrigatórias de senha
chfn	Usado para mudar um nome completo do(a) usante e outras informações
chgpaswd	Usado para atualizar senhas de grupo em modo de lote
chpaswd	Usado para atualizar senhas de usante em modo de lote
chsh	Usado para mudar um shell de login padrão do(a) usante
faillog	É Usado para examinar o registro de falhas de login, configurar um número máximo de falhas antes que uma conta seja bloqueada ou zerar a contagem de falhas
getsubids	É usado para listar os intervalos subordinados de id para um(a) usante
gpaswd	É usado para adicionar e deletar membros(as) e administradores(as) a grupos
groupadd	Cria um grupo com o nome dado
groupdel	Deleta o grupo com o nome dado
groupmod	É usado para modificar o nome ou GID do grupo dado
grpck	Verifica a integridade dos arquivos de grupo <code>/etc/group</code> e <code>/etc/gshadow</code>
grpconv	Cria ou atualiza o arquivo de grupo de sombra a partir do arquivo de grupo normal
grpunconv	Atualiza <code>/etc/group</code> a partir de <code>/etc/gshadow</code> e então deleta o último

login	É usado pelo sistema para permitir usantes logar
newgidmap	É usado para configurar o mapeamento de gid de um espaço de nomes de usante
newgrp	É usado para mudar o GID atual durante uma sessão de login
newuidmap	É usado para configurar o mapeamento de uid de um espaço de nomes de usante
newusers	É usado para criar ou atualizar uma série inteira de contas de usantes
nologin	Exibe uma mensagem dizendo que uma conta não está disponível; é projetado para ser usado como o shell padrão para contas desabilitadas
passwd	É usado para mudar a senha para uma conta de usante ou de grupo
pwck	Verifica a integridade dos arquivos de senha /etc/passwd e /etc/shadow
pwconv	Cria ou atualiza o arquivo de senha de sombra a partir do arquivo de senha normal
pwunconv	Atualiza /etc/passwd a partir de /etc/shadow e então deleta o último
sg	Executa um comando dado enquanto o GID do(a) usante estiver configurado para aquele do grupo dado
su	Executa um shell com IDs substitutos de usante e de grupo
useradd	Cria um(a) usante novo(a) com o nome dado ou atualiza a informação padrão de novo(a) usante
userdel	Deleta a conta especificada de usante
usermod	É usado para modificar o nome de login, identificação de usante (UID), shell, grupo inicial, diretório inicial, entre outras coisas, do(a) usante fornecido(a).
vigr	Edita os arquivos /etc/group ou /etc/gshadow
vipw	Edita os arquivos /etc/passwd ou /etc/shadow
libsubid	biblioteca para lidar com intervalos subordinados de id para usantes e grupos

8.31. Gawk-5.4.1

O pacote Gawk contém aplicativos para manipular arquivos de texto.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 47 MB

8.31.1. Instalação do Gawk

Primeiro, garanta que alguns arquivos desnecessários não sejam instalados:

```
sed -i 's/extras//' Makefile.in
```

Prepare o Gawk para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Instale o pacote:

```
rm -f /usr/bin/gawk-5.4.1
make install
```

O significado do comando:

rm -f /usr/bin/gawk-5.4.1

O sistema de construção não recriará o link rígido `gawk-5.4.1` se ele já existir. Remova-o para garantir que o link rígido anterior instalado na Seção 6.9, “Gawk-5.4.1” seja atualizado aqui.

O processo de instalação já criou **awk** como um link simbólico para **gawk**; crie a página de manual dele como um link simbólico também:

```
ln -sv gawk.1 /usr/share/man/man1/awk.1
```

Se desejado, instale a documentação:

```
install -vDm644 doc/{awkforai.txt,*.eps,pdf,jpg} -t /usr/share/doc/gawk-5.4.1
```

8.31.2. Conteúdo do Gawk

Programas instalados: awk (link para gawk), gawk e awk-5.4.1

Bibliotecas instaladas: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rvarray.so e time.so (todas em /usr/lib/gawk)

Diretórios instalados: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk e /usr/share/doc/gawk-5.4.1

Descrições Curtas

awk Um link para **gawk**

gawk Um aplicativo para manipular arquivos de texto; é a implementação GNU do **awk**

gawk-5.4.1 Um link rígido para **gawk**

8.32. GCC-16.2.0

O pacote GCC contém a GNU Compiler Collection, a qual inclui os compiladores C e C++.

Tempo aproximado de construção: 53 UPC (com os testes)

Espaço em disco exigido: 7,3 GB

8.32.1. Instalação do GCC

Se construir em x86_64, [então] mude o nome padrão de diretório para bibliotecas de 64 bits para “lib”:

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

A documentação do GCC recomenda construir o GCC em um diretório de construção dedicado:

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

Prepare o GCC para compilação:

```
../configure --prefix=/usr \
             LD=ld \
             --enable-languages=c,c++ \
             --enable-default-pie \
             --enable-default-ssp \
             --enable-host-pie \
             --enable-targets=all \
             --disable-multilib \
             --disable-bootstrap \
             --disable-fixincludes \
             --with-system-zlib
```

Nós habilitamos somente C e C++ aqui para economizar o tempo de construção, já que nenhum pacote no LFS e no BLFS exige o GCC para compilar outras linguagens. Posponha `algol68` para Algol 68, `fortran` para Fortran, `go` para Go, `objc` para Objective C, `objc-c++` para Objective C++ e (ou) `m2` para Modula 2 no valor da opção `--enable-languages` se você quiser compilar programas em uma ou mais dessas linguagens com o GCC.

O GCC também suporta as linguagens Ada, COBOL e D. No entanto, isso exigiria algumas dependências que não estão disponíveis no sistema LFS básico e que extrapolariam o escopo deste livro. Leia-se *a documentação de fluxo de desenvolvimento* para detalhes.

O significado dos novos parâmetros de configuração:

`LD=ld`

Esse parâmetro induz o script `configure` a usar o aplicativo `ld` instalado pelo pacote `Binutils`, construído anteriormente neste capítulo, em vez da versão construída cruzadamente, a qual de outra maneira seria usada.

`--disable-bootstrap`

Por padrão, o sistema de construção do GCC autoaprimorará em 3 estágios, a menos que ele seja construído como um compilador cruzado ou esteja sendo compilado cruzadamente. O processo de autoaprimoramento é necessário para robustez, especialmente ao atualizar o GCC para uma nova versão. No LFS, nós estamos usando um método diferente para autoaprimorar o GCC (conforme nós apresentamos nas Observações Técnicas do Conjunto de Ferramentas), de forma que aqui nós não precisamos do processo de autoaprimoramento fornecido pelo sistema de construção e nós o desabilitamos para reduzir significativamente

o tempo de construção. Remova essa opção quando você atualizar o GCC em um sistema LFS completo (em vez de construir o LFS).

`--with-system-zlib`

Essa chave diz ao GCC para vincular à cópia instalada do sistema da biblioteca Zlib, em vez da própria cópia interna dele.

`--enable-targets=all`

Essa chave instrui o GCC a habilitar o suporte à geração de código de 64 bits, mesmo se nós estivermos construindo para um sistema de 32 bits. Ela é necessária para construir o GRUB para UEFI de 64 bits. Essa chave não tem efeito se a construção for para 64 bits.



Nota

PIE (position-independent executables) são aplicativos binários que conseguem ser carregados em qualquer lugar em memória. Sem PIE, o recurso de segurança chamado de ASLR (Address Space Layout Randomization) consegue ser aplicado para as bibliotecas compartilhadas, porém não para os próprios executáveis. Habilitar PIE permite ASLR para os executáveis em adição às bibliotecas compartilhadas e mitiga alguns ataques baseados em endereços fixos de código ou dados sensível(is) nos executáveis.

SSP (Stack Smashing Protection) é uma técnica para garantir que a pilha de parâmetros não está corrompida. A corrupção da pilha consegue, por exemplo, alterar o endereço de retorno de uma sub-rotina, dessa forma transferindo controle para algum código perigoso (existente no aplicativo ou nas bibliotecas compartilhadas; ou injetado pelo(a) atacante de alguma maneira).

Compile o pacote:

```
make
```



Importante

Nesta seção, a suíte de teste para o GCC é considerada importante, porém ela toma um tempo longo. Construtoras(es) de primeira vez são encorajadas(os) a executar a suíte de teste. O tempo para executar os testes pode ser reduzido significativamente adicionando-se `-jx` ao comando **make -k check** abaixo, onde `x` é o número de núcleos da CPU em seu sistema.

O GCC pode precisar de mais espaço de pilha para compilar alguns padrões de código extremamente complexos. Como precaução para as distribuições anfitriãs com um limite de pilha apertado, configure explicitamente o limite rígido do tamanho da pilha como infinito. Na maioria das distribuições anfitriãs (e no sistema LFS final), o limite rígido é infinito por padrão, mas não há mal algum em configurá-lo explicitamente. Não é necessário mudar o limite flexível do tamanho da pilha porque o GCC o configurará automaticamente para um valor apropriado, desde que o valor não exceda o limite rígido:

```
ulimit -s -H unlimited
```

Teste os resultados como um(a) usante não privilegiado(a), porém não pare em erros:

```
chown -R tester .
su tester -c "PATH=$PATH make -k check"
```

Para extrair um sumário dos resultados da suíte de teste, execute:

```
../contrib/test_summary -t
```

Para filtrar somente os sumários, entube a saída gerada por **grep -A7 Summ.**

Os resultados podem ser comparados com aqueles localizados em <https://www.linuxfromscratch.org/lfs/build-logs/13.1/> e <https://gcc.gnu.org/ml/gcc-testresults/>.

Nos testes `gcc.target/i386`, os testes nominados de `auto-init-padding-9.c`, `builtin-memmove-*.c`, `mem{cpy,set}-pr120683-*.c`, `pr111657-1.c`, `pr115102.c`, `pr116896.c`, `pr120881-2a.c` e `pr122343-4a.c` são conhecidos por falharem. O teste nominado de `shift-gf2p8affine-2.c` é conhecido por falhar se o processador não suportar AVX512.

Nos testes `g++.target/i386`, os testes nominados de `memset-pr108585-1{a,b}.C`, `mv{,c}-symbols*.C`, `pr112824-2.c` e `pr116896-1.c` são conhecidos por falharem.

Adicionalmente, os testes `gcc.dg/ipa/pr122458.c`, `gcc.dg/lto/toplevel-*.asm-*` e `gcc.dg/plugin/crash-test-nested-*.c` são conhecidos por falharem. O teste `g++.dg/gomp/deprecate-1.c` é conhecido por falhar algumas vezes.

Os(As) editores(as) do LFS investigaram essas falhas e confirmaram que nenhuma indica um problema crítico. A maioria delas é porque o(a) autor(a) do caso de teste não previu `--enable-default-ssp` ou `--enable-default-pie`.

Um(a)s poucas falhas inesperadas não podem ser evitadas sempre. A menos que os resultados do teste sejam amplamente diferentes daqueles na URL acima, é seguro continuar.

Instale o pacote:

```
make install
```

O diretório de construção do GCC é de propriedade de `tester` agora e a titularidade de propriedade do diretório instalado do cabeçalho (e o conteúdo dele) está incorreto. Mude a titularidade de propriedade para o(a) usante e grupo `root`:

```
chown -v -R root:root $(gcc -print-file-name=include){,-fixed}
```

Crie um link simbólico exigido pelo *FHS* por razões "históricas".

```
ln -svr /usr/bin/cpp /usr/lib
```

Muitos pacotes usam o nome `cc` para chamar o compilador C. Já criamos `cc` como um link simbólico em `gcc-passagem2`; crie a página de manual dele como um link simbólico também:

```
ln -sv gcc.1 /usr/share/man/man1/cc.1
```

Adicione um link simbólico de compatibilidade para habilitar a construção de aplicativos com Link Time Optimization (LTO):

```
ln -sfvr $(gcc -print-prog-name=liblto_plugin.so) /usr/lib/bfd-plugins/
```

Agora que nosso conjunto de ferramentas final está no lugar, é importante certificar-se novamente de que compilação e vinculação funcionarão como esperado. Nós fazemos isso realizando algumas verificações de sanidade:

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

Não deveriam existir erros, e a saída gerada do último comando será (levando em consideração diferenças específicas da plataforma no nome do vinculador dinâmico):

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

Agora certifique-se de que nós estamos configurados para usar os arquivos corretos de iniciação:

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

A saída gerada do último comando deveria ser:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/16.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/16.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/16.2.0/../../../../lib/crtn.o succeeded
```

Dependendo da arquitetura da sua máquina, o acima possivelmente difira ligeiramente. A diferença será o nome do diretório depois de `/usr/lib/gcc`. A coisa importante a se olhar aqui é que o `gcc` tenha encontrado todos os três arquivos `crt*.o` sob o diretório `/usr/lib`.

Verifique se o compilador está procurando os arquivos corretos de cabeçalho:

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

Esse comando deveria retornar a seguinte saída gerada:

```
#include <...> search starts here:
/usr/lib/gcc/x86_64-pc-linux-gnu/16.2.0/include
/usr/local/include
/usr/lib/gcc/x86_64-pc-linux-gnu/16.2.0/include-fixed
/usr/include
```

Novamente, o diretório nomeado depois do teu tripleto alvo possivelmente seja diferente do acima, dependendo da arquitetura do teu sistema.

Agora, verifique se o novo vinculador está sendo usado com os caminhos corretos de procura:

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

Referências para caminhos que tenham componentes com '-linux-gnu' deveriam ser ignoradas, porém, do contrário, a saída gerada do último comando deveria ser:

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

Um sistema de 32 bits possivelmente use uns poucos outros diretórios. Por exemplo, aqui está a saída gerada originária de uma máquina i686:

```
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");
```

Em seguida, tenha certeza de que nós estamos usando a libc correta:

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

A saída gerada do último comando deveria ser:

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

Tenha certeza de que GCC está usando o vinculador dinâmico correto:

```
grep found dummy.log
```

A saída gerada do último comando deveria ser (levando em consideração diferenças específicas da plataforma no nome do vinculador dinâmico):

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

Se a saída gerada não aparecer conforme mostrado acima ou não for recebida, então alguma coisa está seriamente errada. Investigue e refaça as etapas para descobrir onde está o problema e corrija-o. Quaisquer problemas deveriam ser resolvidos antes de se continuar com o processo.

Uma vez que tudo esteja funcionando corretamente, remova os arquivos de teste:

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

Finalmente, mova um arquivo mal colocado:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

8.32.2. Conteúdo do GCC

Programas instalados:	c++, cc (link para gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, gcov-tool e lto-dump
Bibliotecas instaladas:	libasan.{a,so}, libatomic.{a,so}, libgcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libhwasan.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.a, libstdc++exp.a, libstdc++fs.a, libsupc++.a, libtsan.{a,so} e libubsan.{a,so}
Diretórios instalados:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc e /usr/share/gcc-16.2.0

Descrições Curtas

c++	O compilador C++
cc	O compilador C
cpp	O pré-processador C; é usado pelo compilador para expandir as diretivas #include, #define e similares nos arquivos fonte
g++	O compilador C++
gcc	O compilador C
gcc-ar	Um encapsulador em torno de ar que adiciona um plugin à linha de comando. Esse aplicativo é usado somente para adicionar "link time optimization" e não é útil com as opções padrão de construção.
gcc-nm	Um encapsulador em torno de nm que adiciona um plugin à linha de comando. Esse aplicativo é usado somente para adicionar "link time optimization" e não é útil com as opções padrão de construção.
gcc-ranlib	Um encapsulador em torno de ranlib que adiciona um plugin à linha de comando. Esse aplicativo é usado somente para adicionar "link time optimization" e não é útil com as opções padrão de construção.
gcov	Uma ferramenta de testagem de cobertura; usada para analisar aplicativos para determinar onde as otimizações terão o maior efeito
gcov-dump	Ferramenta de despejo de perfil offline gcda e gcno
gcov-tool	Ferramenta de processamento de perfil offline gcda
lto-dump	Ferramenta para despejar arquivos objeto produzidos pelo GCC com LTO habilitado
libasan	A biblioteca de tempo de execução Address Sanitizer
libatomic	Biblioteca de tempo de execução integrada atômica do GCC
libgcc1	Uma biblioteca que permite ao "GDB" fazer uso do "GCC"
libgcc	Contém suporte de tempo de execução para o gcc
libgcov	Essa biblioteca é vinculada a um aplicativo quando o GCC for instruído a habilitar criação de perfil
libgomp	Implementação GNU da API OpenMP para programação paralela de memória compartilhada multiplataforma em C/C++ e Fortran
libhwasan	A biblioteca de tempo de execução do Address Sanitizer assistida por hardware
libitm	A biblioteca de memória transacional GNU
liblsan	A biblioteca de tempo de execução Leak Sanitizer
liblto_plugin	Plugin LTO do GCC permite ao Binutils processar arquivos objeto produzidos pelo GCC com LTO habilitado

<code>libquadmath</code>	API da Biblioteca GCC Quad Precision Math
<code>libssp</code>	Contém rotinas suportantes da funcionalidade de proteção contra esmagamento de pilha do GCC.
<code>libstdc++</code>	A biblioteca padrão C++
<code>libstdc++exp</code>	Biblioteca Experimental de Contratos C++
<code>libstdc++fs</code>	Biblioteca de Sistema de Arquivos ISO/IEC TS 18822:2015
<code>libsupc++</code>	Fornece rotinas suportantes para a linguagem de programação C++
<code>libtsan</code>	A biblioteca de tempo de execução Thread Sanitizer
<code>libubsan</code>	A biblioteca de tempo de execução Undefined Behavior Sanitizer

8.33. Ncurses-6.6

O pacote Ncurses contém bibliotecas para manuseio independente de terminal das telas de caracteres .

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 47 MB

8.33.1. Instalação do Ncurses

Prepare o Ncurses para compilação:

```
./configure --prefix=/usr \
            --mandir=/usr/share/man \
            --with-shared \
            --without-debug \
            --without-normal \
            --with-cxx-shared \
            --enable-pc-files \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

O significado das novas opções de configurar:

--with-shared

Isso faz com que o Ncurses construa e instale bibliotecas C compartilhadas.

--without-normal

Isso evita que o Ncurses construa e instale bibliotecas C estáticas.

--without-debug

Isso evita que o Ncurses construa e instale bibliotecas de depuração.

--with-cxx-shared

Isso faz com que o Ncurses construa e instale vínculos C++ compartilhados. Também evita a construção e instalação de vínculos C++ estáticos.

--enable-pc-files

Essa chave gera e instala arquivos .pc para o pkg-config.

Compile o pacote:

```
make
```

Esse pacote tem uma suíte de teste, entretanto ela só pode ser executada depois que o pacote tiver sido instalado. Os testes residem no diretório `test/`. Veja-se o arquivo `README` naquele diretório para maiores detalhes.

A instalação desse pacote sobrescreverá `libncursesw.so.6.6` no local. Isso possivelmente quebre o processo de shell que esteja usando código e dados oriundo do arquivo da biblioteca. Instale o pacote com `DESTDIR` e substitua corretamente o arquivo da biblioteca usando a opção *--remove-destination* do `cp` (o cabeçalho `curses.h` também é editado para garantir que a ABI de caracteres largos seja usada como o que fizemos em Seção 6.3, “Ncurses-6.6”):

```
make DESTDIR=$PWD/dest install
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i dest/usr/include/curses.h
cp --remove-destination -av dest/* /
```

Muitos aplicativos ainda esperam que o vinculador seja capaz de encontrar bibliotecas Ncurses de caracteres não largos. Engane esses aplicativos para se vincularem a bibliotecas de caracteres largos por meio de links simbólicos (observe que os links `.so` só são seguros com `curses.h` editado para sempre usar a ABI de caracteres largos):

```
for lib in ncurses form panel menu ; do
    ln -sfv lib${lib}w.so /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc /usr/lib/pkgconfig/${lib}.pc
done
```

Finalmente, certifique-se de que aplicativos antigos que procuram por `-lcurses` em tempo de construção ainda sejam construíveis:

```
ln -sfv libncursesw.so /usr/lib/libcurses.so
```

Se desejado, instale a documentação do Ncurses:

```
cp -v -R doc -T /usr/share/doc/ncurses-6.6
```



Nota

As instruções acima não criam bibliotecas Ncurses de caracteres não largos, uma vez que nenhum pacote instalado por compilação a partir dos fontes se vincularia a elas em tempo de execução. Entretanto, os únicos aplicativos somente binário conhecidos que se vinculam à bibliotecas Ncurses de caracteres não largos exigem a versão 5. Se você precisa ter tais bibliotecas, por causa de algum aplicativo somente binário ou para estar conforme com a LSB, construa o pacote novamente com os seguintes comandos:

```
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
```

8.33.2. Conteúdo do Ncurses

Programas instalados:	<code>captinfo</code> (link para <code>tic</code>), <code>clear</code> , <code>infocmp</code> , <code>infotocap</code> (link para <code>tic</code>), <code>ncursesw6-config</code> , <code>reset</code> (link para <code>tset</code>), <code>tabs</code> , <code>tic</code> , <code>toe</code> , <code>tput</code> e <code>tset</code>
Bibliotecas instaladas:	<code>libcurses.so</code> (link simbólico), <code>libform.so</code> (link simbólico), <code>libformw.so</code> , <code>libmenu.so</code> (link simbólico), <code>libmenuw.so</code> , <code>libncurses.so</code> (link simbólico), <code>libncursesw.so</code> , <code>libncurses++w.so</code> , <code>libpanel.so</code> (link simbólico) e <code>libpanelw.so</code> ,
Diretórios instalados:	<code>/usr/share/tabset</code> , <code>/usr/share/terminfo</code> e <code>/usr/share/doc/ncurses-6.6</code>

Descrições Curtas

captinfo	Converte uma descrição <code>termcap</code> em uma descrição <code>terminfo</code>
clear	Limpa a tela, se possível
infocmp	Compara ou imprime descrições <code>terminfo</code>
infotocap	Converte uma descrição <code>terminfo</code> em uma descrição <code>termcap</code>
ncursesw6-config	Fornece informação de configuração para o <code>ncurses</code>
reset	Reinicializa um terminal para valores padrão dele
tabs	Limpa e configura paradas de tabulação em um terminal
tic	O compilador de descrição de entrada do <code>terminfo</code> que traduz um arquivo <code>terminfo</code> do formato fonte para o formato binário necessário para as rotinas de biblioteca do <code>ncurses</code> [Um arquivo <code>terminfo</code> contém informação a respeito dos recursos de um certo terminal].
toe	Lista todos os tipos de terminal disponíveis, dando o nome primário e descrição para cada
tput	Torna os valores de recursos dependentes de terminal disponíveis para o shell; também pode ser usado para reconfigurar ou inicializar um terminal ou informar o nome longo dele
tset	Pode ser usado para inicializar terminais

<code>libncursesw</code>	Contém funções para exibir texto em muitas maneiras complexas em uma tela de terminal; um bom exemplo do uso dessas funções é o menu exibido durante o make menuconfig do núcleo
<code>libncurses++w</code>	Contém vinculamentos C++ para outras bibliotecas nesse pacote
<code>libformw</code>	Contém funções para implementar formulários
<code>libmenuw</code>	Contém funções para implementar menus
<code>libpanelw</code>	Contém funções para implementar painéis

8.34. Sed-4.10

O pacote Sed contém um editor de fluxo.

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 42 MB

8.34.1. Instalação do Sed

Prepare o Sed para compilação:

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

Compile o pacote e gere a documentação HTML:

```
make
make html
```

Para testar os resultados, emita:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Instale o pacote e a documentação dele:

```
make install
install -vDm644 doc/sed.html -t /usr/share/doc/sed-4.10
```

8.34.2. Conteúdo do Sed

Aplicativo instalado: sed

Diretório instalado: /usr/share/doc/sed-4.10

Descrições Curtas

sed Filtra e transforma arquivos de texto em uma passagem única

8.35. Psmisc-23.7

O pacote Psmisc contém aplicativos para exibir informação a respeito de processos em execução.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 6,8 MB

8.35.1. Instalação do Psmisc

Prepare Psmisc para compilação:

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

Compile o pacote:

```
make
```

Para executar a suíte de teste, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.35.2. Conteúdo do Psmisc

Programas instalados: fuser, killall, peekfd, prtstat, pslog, pstree e pstree.x11 (link para pstree)

Descrições Curtas

fuser	Informa os IDs de Processos (PIDs) de processos que usam os arquivos ou sistemas de arquivos dados
killall	Mata processos pelo nome; envia um sinal para todos os processos executando quaisquer dos comandos dados
peekfd	Dê uma olhada nos descritores de arquivo de um processo em execução, dado seu PID
prtstat	Imprime informação a respeito de um processo
pslog	Informa o caminho atual de registros de um processo
pstree	Exibe processos em execução como uma árvore
pstree.x11	O mesmo que pstree , exceto que ele espera por confirmação antes de sair

8.36. Gettext-1.0

O pacote Gettext contém utilitários para internacionalização e localização. Eles permitem que programas sejam compilados com Suporte ao Idioma Nativo (Native Language Support - NLS), habilitando-os a emitir mensagens no idioma nativo do(a) usante.

Tempo aproximado de construção: 2,1 UPC

Espaço em disco exigido: 448 MB

8.36.1. Instalação do Gettext

Prepare o Gettext para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/gettext-1.0
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.36.2. Conteúdo do Gettext

Programas instalados:	autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin e xgettext
Bibliotecas instaladas:	libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so e preloadable_libintl.so
Diretórios instalados:	/usr/lib/gettext, /usr/share/doc/gettext-1.0, /usr/share/gettext e /usr/share/gettext-1.0

Descrições Curtas

autopoint	Copia arquivos de infraestrutura padrão do Gettext para um pacote fonte
envsubst	Substitui variáveis de ambiente em sequências de caracteres de formato de shell
gettext	Traduz uma mensagem natural de idioma para o idioma do(a) usante procurando a tradução em um catálogo de mensagens
gettext.sh	Primariamente serve como uma biblioteca de função de shell para o gettext
gettextize	Copia todos os arquivos padrão do Gettext para o diretório de nível superior fornecido de um pacote para começar a internacionalizá-lo
msgattrib	Filtra as mensagens de um catálogo de tradução de acordo com os atributos delas e manipula os atributos
msgcat	Concatena e mescla os arquivos .po fornecidos
msgcmp	Compara dois arquivos .po para verificar se ambos contém o mesmo conjunto de sequências de caracteres de msgid

msgcomm	Encontra as mensagens que são comuns aos arquivos .po fornecidos
msgconv	Converte um catálogo de tradução para uma codificação de caracteres diferente
msgen	Cria um catálogo de tradução em inglês
msgexec	Aplica um comando a todas as traduções de um catálogo de tradução
msgfilter	Aplica um filtro a todas as traduções de um catálogo de tradução
msgfmt	Gera um catálogo de mensagem binária a partir de um catálogo de tradução
msggrep	Extraí todas as mensagens de um catálogo de tradução que correspondem a um determinado padrão ou pertencem a alguns arquivos fonte fornecidos
msginit	Cria um novo arquivo .po, inicializando a meta informação com valores oriundos do ambiente do(a) usante
msgmerge	Combina duas traduções cruas em arquivo único
msgunfmt	Descompila um catálogo de mensagem binário em texto de tradução cru
msguniq	Unifica traduções duplicadas em um catálogo de tradução
ngettext	Exibe traduções no idioma nativo de uma mensagem textual cuja forma gramatical depende de um número
recode-sr-latin	Re-codifica texto sérvio do cirílico para alfabeto latino
xgettext	Extraí as linhas de mensagem traduzíveis dos arquivos fonte fornecidos para fazer o primeiro modelo de tradução
libasprintf	Define a classe <i>autosprintf</i> , que torna as rotinas de saída gerada formatada em C utilizáveis em aplicativos C++, para uso com as sequências de caracteres <i><string></i> e os fluxos <i><iostream></i>
libgettextlib	Contém rotinas comuns usadas pelos vários aplicativos do Gettext; elas não são destinadas para uso geral
libgettextpo	Usada para escrever aplicativos especializados que processam arquivos .po; essa biblioteca é usada quando os aplicativos padrão embarcados com o Gettext (tais como msgcomm , msgcmp , msgattrib e msgen) não bastarão
libgettextsrc	Fornece rotinas comuns usadas pelos vários aplicativos do Gettext; elas não são destinadas para uso geral
libtextstyle	Biblioteca de estilização de texto
preloadable_libintl	Uma biblioteca, destinada a ser usada por LD_PRELOAD que auxilia a libintl a registrar mensagens não traduzidas

8.37. Bison-3.8.2

O pacote Bison contém um gerador de analisador.

Tempo aproximado de construção: 2,1 UPC

Espaço em disco exigido: 63 MB

8.37.1. Instalação do Bison

Prepare o Bison para compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.8.2
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.37.2. Conteúdo do Bison

Programas instalados: bison e yacc
Biblioteca instalada: liby.a
Diretório instalado: /usr/share/bison

Descrições Curtas

bison	Gera, a partir de uma série de regras, um aplicativo para analisar a estrutura de arquivos de texto; Bison é uma substituição ao Yacc (Yet Another Compiler Compiler)
yacc	Um encapsulador para bison , destinado a aplicativos que ainda chamam yacc em vez de bison ; ele chama bison com a opção -y
liby	A biblioteca Yacc contendo implementações de funções compatíveis com Yacc yyerror e main; essa biblioteca normalmente não é muito útil, mas POSIX a exige

8.38. Grep-3.12

O pacote Grep contém aplicativos para procura ao longo do conteúdo de arquivos.

Tempo aproximado de construção: 0,5 UPC

Espaço em disco exigido: 48 MB

8.38.1. Instalação do Grep

Primeiro, remova um aviso a respeito de usar egrep e fgrep que induz os testes em alguns pacotes a falharem:

```
sed -i "s/echo/#echo/" src/egrep.sh
```

Prepare o Grep para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.38.2. Conteúdo do Grep

Programas instalados: egrep, fgrep e grep

Descrições Curtas

- egrep** Imprime linhas correspondentes a uma expressão regular estendida. Isso está obsoleto; use **grep -E** em vez disso
- fgrep** Imprime linhas correspondentes a uma lista de sequências de caracteres fixas. Isso está obsoleto; use **grep -F** em vez disso
- grep** Imprime linhas correspondentes a expressão regular básica

8.39. Bash-5.3

O pacote Bash contém o Bourne-Again Shell.

Tempo aproximado de 1,4 UPC

construção:

Espaço em disco exigido: 56 MB

8.39.1. Instalação do Bash

Prepare o Bash para compilação:

```
./configure --prefix=/usr \
            --without-bash-malloc \
            --with-installed-readline \
            --docdir=/usr/share/doc/bash-5.3
```

O significado da nova opção do configure:

--with-installed-readline

Essa opção diz ao Bash para usar a biblioteca `readline` que já está instalada no sistema em vez de usar a própria versão dele da `readline`.

Compile o pacote:

```
make
```

Pule para “Instale o pacote” se não executar a suíte de teste.

Para preparar os testes, garanta que o(a) usante `tester` consiga escrever na árvore dos fontes:

```
chown -R tester .
```

A suíte de teste desse pacote é projetada para ser executada como um(a) usante não `root` que é proprietário(a) do terminal conectado à entrada padrão. Para satisfazer a exigência, gere um novo pseudo terminal usando o `Expect` e execute os testes como o(a) usante `tester`:

```
LC_ALL=C.UTF-8 su -s /usr/bin/expect tester << "EOF"
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

A suíte de teste usa o **diff** para detectar a diferença entre a saída gerada do script de teste e a saída gerada esperada. Qualquer saída gerada oriunda do **diff** (prefixada com `<` e `>`) indica uma falha de teste, a menos que exista uma mensagem dizendo que a diferença pode ser ignorada. O teste chamado `run-builtins` é conhecido por falhar em algumas distribuições anfitriãs com uma diferença nas linhas 479 e 480 da saída gerada. Alguns outros testes precisam dos locais `zh_TW.BIG5` e `ja_JP.SJIS`; eles são conhecidos por falhar, a menos que esses locais estejam instalados.

Instale o pacote:

```
make install
```

Execute o aplicativo recém compilado **bash** (substituindo o que está sendo executado atualmente):

```
exec /usr/bin/bash --login
```

8.39.2. Conteúdo do Bash

Programas instalados: `bash`, `bashbug` e `sh` ([link para bash](#))

Diretório instalado: `/usr/include/bash`, `/usr/lib/bash` e `/usr/share/doc/bash-5.3`

Descrições Curtas

- bash** Um interpretador de comandos vastamente usado; ele realiza muitos tipos de expansões e substituições sobre uma dada linha de comando antes de executá-la, portanto fazendo desse interpretador uma ferramenta poderosa
- bashbug** Um conjunto de comandos sequenciados de shell para ajudar o(a) usante a compor e enviar informes padrão formatados de defeitos concernentes ao **bash**
- sh** Um link simbólico para o aplicativo **bash**; quando invocado como **sh**, o **bash** tenta imitar o comportamento de inicialização de versões históricas do **sh** o mais próximo possível, enquanto conformante ao padrão POSIX também

8.40. Libtool-2.6.2

O pacote Libtool contém o script de suporte à biblioteca genérica GNU. Ele torna o uso de bibliotecas compartilhadas mais simples com uma interface consistente, portátil.

Tempo aproximado de 0,6 UPC

construção:

Espaço em disco exigido: 50 MB

8.40.1. Instalação do Libtool

Prepare Libtool para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

Remova uma biblioteca estática útil somente para a suíte de teste:

```
rm -fv /usr/lib/libltdl.a
```

8.40.2. Conteúdo do Libtool

Programas instalados: libtool e libtoolize

Bibliotecas instaladas: libltdl.so

Diretórios instalados: /usr/include/libltdl e /usr/share/libtool

Descrições Curtas

libtool Fornece serviços generalizados de suporte à construção de biblioteca

libtoolize Fornece uma maneira padrão de adicionar suporte **libtool** a um pacote

libltdl Esconde as várias dificuldades do abrir dinamicamente bibliotecas carregadas

8.41. GDBM-1.26

O pacote GDBM contém o GNU Database Manager. Ele é uma biblioteca de funções de base de dados que usa hash extensível e funciona semelhante ao dbm UNIX padrão. A biblioteca fornece primitivos para armazenar pares de chave/dados, pesquisar e recuperar os dados pela sua chave deles e deletar uma chave junto com os dados dela.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 13 MB

8.41.1. Instalação do GDBM

Prepare GDBM para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --enable-libgdbm-compat
```

O significado da opção de configure:

--enable-libgdbm-compat

Essa chave habilita construir a biblioteca de compatibilidade libgdbm. Alguns pacotes fora do LFS possivelmente exijam as rotinas DBM mais antigas que ela fornece.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.41.2. Conteúdo do GDBM

Programas instalados: gdbm_dump, gdbm_load e gdbmtool

Bibliotecas instaladas: libgdbm.so e libgdbm_compat.so

Descrições Curtas

gdbm_dump	Despeja uma base de dados GDBM para um arquivo
gdbm_load	Recria uma base de dados GDBM a partir de um arquivo de despejo
gdbmtool	Testa e modifica uma base de dados GDBM
libgdbm	Contém funções para manipular uma base de dados com hash
libgdbm_compat	Biblioteca de compatibilidade contendo funções DBM mais antigas

8.42. Gperf-3.3

Gperf gera uma função de hash perfeita a partir de um conjunto de chaves.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 12 MB

8.42.1. Instalação do Gperf

Prepare Gperf para compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.3
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.42.2. Conteúdo do Gperf

Aplicativo instalado: gperf

Diretório instalado: /usr/share/doc/gperf-3.3

Descrições Curtas

gperf Gera um hash perfeito a partir de um conjunto de chaves

8.43. Expat-2.8.3

O pacote Expat contém uma biblioteca C orientada a fluxo para analisar XML.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 12 MB

8.43.1. Instalação do Expat

Prepare Expat para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.8.3
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

Se desejado, instale a documentação:

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.8.3
```

8.43.2. Conteúdo do Expat

Aplicativo instalado:	xmlwf
Bibliotecas instaladas:	libexpat.so
Diretório instalado:	/usr/share/doc/expat-2.8.3

Descrições Curtas

xmlwf	É um utilitário não validador para verificar se documentos XML estão bem formados ou não
libexpat	Contém funções de API para analisar XML

8.44. Inetutils-2.8

O pacote Inetutils contém aplicativos para operação interativa básica de dispositivos via rede de comunicação.

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 38 MB

8.44.1. Instalação do Inetutils

Primeiro, faça a construção do pacote com gcc-14.1 ou posterior:

```
sed -i 's/def HAVE_TERMCAP_TGETENT/ 1/' telnet/telnet.c
```

Prepare Inetutils para compilação:

```
./configure --prefix=/usr \
            --bindir=/usr/bin \
            --localstatedir=/var \
            --disable-logger \
            --disable-whois \
            --disable-rcp \
            --disable-rexec \
            --disable-rlogin \
            --disable-rsh \
            --disable-servers
```

O significado das opções do configure:

--disable-logger

Essa opção impede que o Inetutils instale o aplicativo **logger**, o qual é usado por scripts para passar mensagens para o System Log Daemon. Não o instale, pois o Util-linux instala uma versão mais recente.

--disable-whois

Essa opção desabilita a construção do cliente **whois** do Inetutils, o qual está desatualizado. Instruções para um cliente **whois** melhor estão no livro BLFS.

*--disable-r**

Esses parâmetros desabilitam a construção de aplicativos obsoletos que não deveriam ser usados devido a problemas de segurança. As funções fornecidas por esses aplicativos podem ser fornecidas pelo pacote openssh no livro BLFS.

--disable-servers

Isso desabilita a instalação dos vários servidores de rede de comunicação incluídos como parte do pacote Inetutils. Esses servidores são considerados inadequados em um sistema LFS básico. Alguns são inseguros por natureza e só são considerados seguros em redes de comunicação confiáveis. Observe que substituições melhores estão disponíveis para muitos desses servidores.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

Mova um aplicativo para o local adequado:

```
mv -v /usr/{,s}bin/ifconfig
```

8.44.2. Conteúdo do Inetutils

Programas instalados: `dnsdomainname`, `ftp`, `ifconfig`, `hostname`, `ping`, `ping6`, `talk`, `telnet`, `tftp` e `traceroute`

Descrições Curtas

<code>dnsdomainname</code>	Mostra o nome de domínio DNS do sistema
<code>ftp</code>	É o aplicativo do protocolo de transferência de arquivos
<code>hostname</code>	Informa ou configura o nome do dispositivo
<code>ifconfig</code>	Gerencia interfaces da rede de comunicação
<code>ping</code>	Envia pacotes de solicitação de echo e informa quanto tempo as respostas demoram
<code>ping6</code>	Uma versão do <code>ping</code> para redes de comunicação IPv6
<code>talk</code>	É usado para conversar com outro(a) usante
<code>telnet</code>	Uma interface para o protocolo TELNET
<code>tftp</code>	Um aplicativo de transferência de arquivos trivial
<code>traceroute</code>	Traça a rota que seus pacotes fazem a partir do dispositivo no qual você está trabalhando para outro dispositivo em uma rede de comunicação, mostrando todos os saltos intermediários (gateways) ao longo do caminho

8.45. Less-704

O pacote Less contém um visualizador de arquivos de texto.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 17 MB

8.45.1. Instalação do Less

Prepare Less para compilação:

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

O significado das opções do configure:

`--sysconfdir=/etc`

Essa opção diz aos aplicativos criados pelo pacote para procurarem em `/etc` pelos arquivos de configuração.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.45.2. Conteúdo do Less

Programas instalados: less, lessecho e lesskey

Descrições Curtas

less	Um visualizador de arquivos ou paginador; ele exibe o conteúdo do arquivo fornecido, permitindo que o(a) usante role, encontre sequências de caracteres e pule para marcas
lessecho	Necessário para expandir meta caracteres, tais como <code>*</code> e <code>?</code> , em nomes de arquivos em sistemas Unix
lesskey	Usado para especificar os atalhos de tecla para o less

8.46. Perl-5.44.0

O pacote Perl contém o Practical Extraction and Report Language.

Tempo aproximado de construção: 1,3 UPC

Espaço em disco exigido: 261 MB

8.46.1. Instalação do Perl

Essa versão do Perl constrói os módulos `Compress::Raw::Zlib` e `Compress::Raw::BZip2`. Por padrão, Perl usará uma cópia interna dos fontes para a construção. Emita o seguinte comando de modo que o Perl usará as bibliotecas instaladas no sistema:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Para ter controle completo sobre a maneira como o Perl é configurado, você pode remover as opções “-des” do comando seguinte e escolher manualmente a maneira como esse pacote é construído. Alternativamente, use o comando exatamente como mostrado abaixo para usar os padrões que o Perl detecta automaticamente:

```
sh Configure -des \
-D prefix=/usr \
-D vendorprefix=/usr \
-D privlib=/usr/lib/perl5/5.44/core_perl \
-D archlib=/usr/lib/perl5/5.44/core_perl \
-D sitelib=/usr/lib/perl5/5.44/site_perl \
-D sitearch=/usr/lib/perl5/5.44/site_perl \
-D vendorlib=/usr/lib/perl5/5.44/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.44/vendor_perl \
-D man1dir=/usr/share/man/man1 \
-D man3dir=/usr/share/man/man3 \
-D pager="/usr/bin/less -isR" \
-D useshrplib \
-D usethreads
```

O significado das novas opções do "Configure":

`-D pager="/usr/bin/less -isR"`

Isso garante que o `less` seja usado em vez do `more`.

`-D man1dir=/usr/share/man/man1 -D man3dir=/usr/share/man/man3`

Uma vez que o Groff ainda não está instalado, **Configure** não criará páginas de manual para o Perl. Esses parâmetros substituem esse comportamento.

`-D usethreads`

Constrói O Perl com suporte para camadas.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
TEST_JOBS=$(nproc) make test_harness
```

Instale o pacote e limpe:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.46.2. Conteúdo do Perl

Programas instalados:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.44.0 (link rígido para perl), perlbug, perldoc, perlvp, perlthanks (link rígido para perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp e zipdetails
Bibliotecas instaladas:	Muitas das quais não podem ser todas listadas aqui
Diretório instalado:	/usr/lib/perl5

Descrições Curtas

corelist	Uma estrutura de interação direta com o(a) usante por linha de comando para Module::CoreList
cpan	Interage com a Comprehensive Perl Archive Network (CPAN) a partir da linha de comando
enc2xs	Constrói uma extensão do Perl para o módulo Encode a partir tanto de Mapeamentos de Caracteres Unicode quanto de Arquivos de Codificação da Tcl
encguess	Advinha o tipo de codificação de um ou vários arquivos
h2ph	Converte arquivos de cabeçalho C .h para arquivos de cabeçalho do Perl .ph
h2xs	Converte arquivos de cabeçalho C .h para extensões do Perl
instmodsh	Script de shell para examinar módulos do Perl instalados; consegue criar um tarball a partir de um módulo instalado
json_pp	Converte dados entre certos formatos de entrada gerada e de saída gerada
libnetcfg	Pode ser usado para configurar o módulo do Perl libnet
perl	Combina alguns dos melhores recursos do C, sed , awk e sh em uma linguagem canivete suíço única
perl5.44.0	Um link rígido para perl
perlbug	Usado para gerar informes de defeitos a respeito do Perl ou dos módulos que vem como ele, e enviá-los por correio
perldoc	Exibe um pedaço da documentação em formato pod que está embutida na árvore de instalação do Perl ou em um script do Perl
perlvp	O Procedimento de Verificação de Instalação do Perl; pode ser usado para verificar se o Perl e as bibliotecas dele foram instalados corretamente
perlthanks	Usado para gerar mensagens de agradecimento para enviar para os(as) desenvolvedores(as) do Perl
piconv	Uma versão Perl do conversor de codificação de caracteres iconv
pl2pm	Uma ferramenta rudimentar para converter arquivos Perl4 .pl para módulos Perl5 .pm
pod2html	Converte arquivos do formato pod para o formato HTML
pod2man	Converte dados pod para entrada gerada formatada *roff
pod2text	Converte dados pod para texto formatado ASCII
pod2usage	Imprime mensagens de uso a partir de documentos pod embutidos em arquivos
podchecker	Verifica a sintaxe de arquivos de documentação do formato pod
podselect	Exibe seções selecionadas da documentação pod
prove	Ferramenta de linha de comando para executar testes contra o módulo Test::Harness
ptar	Um aplicativo similar ao tar escrito em Perl
ptardiff	Um aplicativo Perl que compara um arquivamento extraído com um não extraído

ptargrep	Um aplicativo Perl que aplica correspondência de padrão ao conteúdo de arquivos em um arquivamento tar
shasum	Imprime ou verifica somas de verificação SHA
splain	É usado para forçar diagnósticos verbosos de aviso em Perl
xsubpp	Converte código Perl XS em código C
zipdetails	Exibe detalhes a respeito da estrutura interna de um arquivo Zip

8.47. Autoconf-2.73

O pacote Autoconf contém aplicativos para produzir scripts de shell que conseguem configurar automaticamente código fonte.

Tempo aproximado de construção: menos que 0,1 UPC (cerca de 0,4 UPC com os testes)

Espaço em disco exigido: 25 MB

8.47.1. Instalação do Autoconf

Prepare Autoconf para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.47.2. Conteúdo do Autoconf

Programas instalados: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames
Diretório instalado: /usr/share/autoconf

Descrições Curtas

autoconf	Produz scripts de shell que configuram automaticamente pacotes de código fonte de aplicativos para adaptar a muitos tipos de sistemas semelhantes a Unix; os scripts de configuração que ele produz são independentes—executá-los não exige o aplicativo autoconf
autoheader	Uma ferramenta para criar arquivos de modelo de declarações <i>#define</i> da C para o configure usar
autom4te	Um encapsulador para o processador de macro M4
autoreconf	Automaticamente executa autoconf , autoheader , aclocal , automake , gettextize e libtoolize na ordem correta para economizar tempo quando mudanças são feitas para arquivos de modelo autoconf e automake
autoscan	Ajuda a criar um arquivo <code>configure.in</code> para um pacote de aplicativos; ele examina os arquivos fonte em uma árvore de diretórios, procurando neles por problemas comuns de portabilidade e cria um arquivo <code>configure.scan</code> que serve como um arquivo <code>configure.in</code> preliminar para o pacote
autoupdate	Modifica um arquivo <code>configure.in</code> que ainda chama macros autoconf pelos nomes antigos delas para usar os nomes atuais de macro
ifnames	Ajuda ao escrever arquivos <code>configure.in</code> para um pacote de software; ele imprime os identificadores que o pacote usa em condicionais de pré-processador C [Se um pacote já tiver sido configurado para ter alguma portabilidade, esse programa consegue ajudar a determinar o que o configure precisa verificar. Ele também consegue preencher lacunas em um arquivo <code>configure.in</code> gerado pelo autoscan].

8.48. Automake-1.18.1

O pacote Automake contém aplicativos para gerar arquivos Make para uso com o Autoconf.

Tempo aproximado de construção: menos que 0,1 UPC (cerca de 1,1 UPC com os testes)

Espaço em disco exigido: 124 MB

8.48.1. Instalação do Automake

Prepare Automake para compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.18.1
```

Compile o pacote:

```
make
```

Usar-se quatro tarefas paralelas acelera os testes, mesmo em sistemas com menos núcleos lógicos, devido a atrasos internos em testes individuais. Para testar os resultados, emita:

```
make -j$(($(nproc)>4?$(nproc):4)) check
```

Substitua "\$((...))" pelo número de núcleos lógicos que você deseja usar se não quiser usar todos.

Instale o pacote:

```
make install
```

8.48.2. Conteúdo do Automake

Programas instalados: aclocal, aclocal-1.18 (vinculado rigidamente com aclocal), automake e automake-1.18 (vinculado rigidamente com automake)

Diretórios instalados: /usr/share/aclocal-1.18, /usr/share/automake-1.18 e /usr/share/doc/automake-1.18.1

Descrições Curtas

aclocal	Gera arquivos aclocal.m4 baseados no conteúdo dos arquivos configure.in
aclocal-1.18	Um link rígido para aclocal
automake	Uma ferramenta para gerar automaticamente arquivos Makefile.in a partir de arquivos Makefile.am [Para criar todos os arquivos Makefile.in para um pacote, execute esse aplicativo no diretório de nível superior. Escaneando o arquivo configure.in, ele automaticamente encontra cada arquivo Makefile.am apropriado e gera o arquivo Makefile.in correspondente].
automake-1.18	Um link rígido para automake

8.49. OpenSSL-4.0.1

O pacote OpenSSL contém ferramentas de gerenciamento e bibliotecas relacionadas à criptografia. Essas são úteis para fornecer funções criptográficas para outros pacotes, tais como o OpenSSH, aplicativos de correio eletrônico e navegadores de rede (para acessar sítios HTTPS).

Tempo aproximado de construção: 1,9 UPC

Espaço em disco exigido: 1,0 GB

8.49.1. Instalação do OpenSSL

Prepare OpenSSL para compilação:

```
./config --prefix=/usr \
--openssldir=/etc/ssl \
--libdir=lib \
shared \
zlib-dynamic
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make test
```

Por padrão, a suíte de teste usa todos os núcleos de CPU disponíveis. Se você quiser usar menos núcleos, anteponha `HARNESS_JOBS=<N>`, com N substituído pelo número de núcleos lógicos de CPU que deseja usar.

Um teste, `30-test_afalg.t`, é conhecido por falhar se o núcleo do anfitrião não tiver `CONFIG_CRYPTO_USER_API_SKCIPHER` habilitado ou não tiver quaisquer opções fornecendo um AES com implementação CBC (por exemplo, a combinação de `CONFIG_CRYPTO_AES` e `CONFIG_CRYPTO_CBC`; ou `CONFIG_CRYPTO_AES_NI_INTEL` se a CPU suportar AES-NI) habilitado. Se falhar, pode ser seguramente ignorado.

Instale o pacote (configurar-se uma `INSTALL_LIBS` vazia impede a instalação das bibliotecas estáticas):

```
make INSTALL_LIBS= MANSUFFIX=ssl install
```

Adicione a versão ao nome de diretório de documentação, para ser consistente com outros pacotes:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-4.0.1
```

Se desejado, instale alguma documentação adicional:

```
cp -vfr doc/* /usr/share/doc/openssl-4.0.1
```



Nota

Você deveria atualizar o OpenSSL quando uma versão nova que corrige vulnerabilidades for anunciada. Desde o OpenSSL 3.0.0, o esquema de versionamento do OpenSSL segue o formato MAIOR.MENOR.REMENDO. Compatibilidade de API/ABI é garantida para o mesmo número de versão MAIOR. Por causa de que o LFS instala somente as bibliotecas compartilhadas, não existe necessidade de recompilar pacotes que se vinculem à `libcrypto.so` ou à `libssl.so` *quando atualizar para uma versão com o mesmo número de versão MAIOR*.

Entretanto, quaisquer aplicativos em execução vinculados àquelas bibliotecas precisam ser parados e reiniciados. Leiam-se as entradas relacionadas em Seção 8.2.1, “Problemas de Atualização” para detalhes.

8.49.2. Conteúdo do OpenSSL

Programas instalados: c_rehash e openssl
Bibliotecas instaladas: libcrypto.so e libssl.so
Diretórios instalados: /etc/ssl, /usr/include/openssl, /usr/lib/engines e /usr/share/doc/openssl-4.0.1

Descrições Curtas

c_rehash	é um script Perl que escaneia todos os arquivos em um diretório e adiciona links simbólicos para os valores de hash deles. O uso do c_rehash é considerado obsoleto e deveria ser substituído pelo comando openssl rehash
openssl	é uma ferramenta de linha de comando para usar as várias funções criptográficas da biblioteca de criptografia do "OpenSSL" a partir do "shell". Ela pode ser usada para várias funções que estão documentadas em " <i>openssl(1)</i> "
libcrypto.so	implementa um intervalo amplo de algoritmos criptográficos usados em vários padrões da Internet. Os serviços fornecidos por essa biblioteca são usados pelas implementações OpenSSL do SSL, TLS e S/MIME e eles também tem sido usados para implementar OpenSSH, OpenPGP e outros padrões criptográficos
libssl.so	implementa o protocolo "Transport Layer Security" ("TLS v1"). Ela fornece uma "API" rica, documentação a respeito da qual pode ser encontrada em " <i>ssl(7)</i> "

8.50. Libelf originário do Elfutils-0.195

Libelf é uma biblioteca para lidar com arquivos ELF (Executable and Linkable Format).

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 159 MB

8.50.1. Instalação do Libelf

Libelf é parte do pacote elfutils-0.195. Use o arquivo elfutils-0.195.tar.bz2 como o tarball fonte.

Prepare Libelf para compilação:

```
./configure --prefix=/usr \
            --disable-debuginfod \
            --enable-libdebuginfod=dummy
```

Compile somente o Libelf:

```
make -C lib
make -C libelf
```

Para testar os resultados, emita:

```
make -k check
```

O comando acima constrói os utilitários que são parte do pacote, bem como os testes de unidade.

Instale somente o Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.50.2. Conteúdo do Libelf

Biblioteca instalada: libelf.so

Diretório instalado: /usr/include/elfutils

Descrições Curtas

`libelf.so` Contém funções de API para lidar com arquivos objeto ELF

8.51. Libffi-3.8.0

A biblioteca Libffi fornece uma interface de programação portátil e de alto nível para várias convenções de chamada. Isso permite a um(a) programador(a) chamar qualquer função especificada por uma descrição de interface de chamada em tempo de execução.

FFI significa Foreign Function Interface. Uma FFI permite a um aplicativo escrito em uma linguagem chamar um aplicativo escrito em outra linguagem. Especificamente, Libffi consegue fornecer uma ponte entre um interpretador, como Perl ou Python, e sub-rotinas da biblioteca compartilhada escrita em C, ou em C++.

Tempo aproximado de 1,7 UPC

construção:

Espaço em disco exigido: 12 MB

8.51.1. Instalação do Libffi



Nota

Assim como o GMP, o Libffi constrói com otimizações específicas para o processador em uso. Se construir para outro sistema, mude o valor do parâmetro `--with-gcc-arch=` no comando a seguir para um nome de arquitetura totalmente implementado por **ambos** a CPU do anfitrião e a CPU naquele sistema. Se isso não for feito, todos os aplicativos que se vincularem a `libffi` deflagrarão Erros de Operação Ilegal. Se você não conseguir descobrir um valor seguro para ambas as CPUs, substitua o parâmetro por `--without-gcc-arch` para produzir uma biblioteca genérica.

Prepare Libffi para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --with-gcc-arch=native
```

O significado da opção de configure:

`--with-gcc-arch=native`

Garante que o GCC otimize para o sistema atual. Se isso não for especificado, [então] o sistema é presumido e o código gerado possivelmente não esteja correto. Se o código gerado será copiado de um sistema nativo para um sistema menos capaz, [então] use o sistema menos capaz como um parâmetro. Para detalhes acerca de tipos alternativos de sistema, veja-se *as opções de x86 no manual do GCC*.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.51.2. Conteúdo do Libffi

Biblioteca instalada: libffi.so

Descrições Curtas

`libffi` Contém as funções de API da interface de função externa

8.52. Sqlite-3530400

O pacote Sqlite é uma biblioteca de software que implementa um mecanismo de base de dados SQL transacional, autocontido, sem servidor e com configuração zero.

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 128 MB

8.52.1. Instalação do Sqlite

Desempacote a documentação:

```
python3 -m zipfile -e ../sqlite-doc-3530400.zip .
```

Prepare o Sqlite para compilação com:

```
./configure --prefix=/usr \
            --disable-static \
            --enable-fts{4,5} \
            CPPFLAGS="-D SQLITE_ENABLE_COLUMN_METADATA=1 \
                    -D SQLITE_ENABLE_UNLOCK_NOTIFY=1 \
                    -D SQLITE_ENABLE_DBSTAT_VTAB=1 \
                    -D SQLITE_SECURE_DELETE=1"
```

O significado das opções do configure:

`--enable-fts{4,5}`

Essas chaves habilitam suporte para as versões 4 e 5 da extensão de pesquisa de texto completo (PTC).

`CPPFLAGS="-D SQLITE_ENABLE_COLUMN_METADATA=1 ...`

Alguns aplicativos exigem que essas opções estejam ativadas. A única maneira de fazer isso é a de incluí-las em CFLAGS ou CPPFLAGS. Nós usamos esse último, de modo que o valor padrão (ou qualquer valor configurado pelo(a) usante) de CFLAGS não seja afetado. Para mais informações acerca do que pode ser especificado, veja-se <https://www.sqlite.org/compile.html>.

Compile o pacote:

```
make LDFLAGS.rpath=""
```

A opção `LDFLAGS.rpath=""` impede a codificação rígida de caminhos de busca de bibliotecas (rpath) na biblioteca compartilhada. Esse pacote não precisa do rpath para uma instalação no local padrão, e o rpath pode, às vezes, causar efeitos indesejados ou até mesmo problemas de segurança.

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

Se desejado, instale a documentação:

```
cp -v -R sqlite-doc-3530400 -T /usr/share/doc/sqlite-3.53.4
```

8.52.2. Conteúdo do Sqlite

Programas instalados:	sqlite3
Bibliotecas instaladas:	libsqlite3.so
Diretórios instalados:	/usr/share/doc/sqlite-3.53.4

Descrições Curtas

sqlite3 é uma estrutura de interação direta com o(a) usante baseada em terminal para a biblioteca SQLite que pode avaliar consultas interativamente e exibir os resultados

`libsqlite3.so` contém as funções da API do SQLite

8.53. mpdecimal-4.0.1

O pacote mpdecimal contém bibliotecas rápidas C/C++ para aritmética de ponto flutuante decimal de precisão arbitrária com arredondamento correto.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 4,4 MB

8.53.1. Instalação do mpdecimal

Prepare mpdecimal para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpdecimal-4.0.1
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check_local
```

Instale o pacote:

```
make install
```

8.53.2. Conteúdo do mpdecimal

Bibliotecas instaladas: libmpdec.so e libmpdec++.so

Diretório instalado: /usr/share/doc/mpdecimal-4.0.1

Descrições Curtas

libmpdec	Contém funções de aritmética de ponto flutuante decimal com precisão arbitrária para programas em C
libmpdec++	Contém funções de aritmética de ponto flutuante decimal com precisão arbitrária para programas em C++

8.54. Python-3.14.7

O pacote Python 3 contém o ambiente de desenvolvimento do Python. Ele é útil para programação orientada a objeto, escrita de scripts, prototipagem de aplicativos grandes e desenvolvimento de aplicações inteiras. Python é uma linguagem interpretada de computador.

Tempo aproximado de construção: 2,7 UPC

Espaço em disco exigido: 509 MB

8.54.1. Instalação do Python 3

Primeiro, aplique um remendo para compatibilidade com OpenSSL 4:

```
patch -Np1 -i ../Python-3.14.7-openssl_4-1.patch
```

Prepare o Python para compilação:

```
./configure --prefix=/usr \
            --enable-shared \
            --with-system-expat \
            --enable-optimizations \
            --without-static-libpython
```

O significado das opções do configure:

--with-system-expat

Essa chave habilita vinculação contra a versão de sistema do Expat.

--enable-optimizations

Essa chave habilita etapas extensivas, porém consumidoras de tempo, de otimização. O interpretador é construído duas vezes; testes realizados na primeira construção são usados para melhorar a versão otimizada final.

Compile o pacote:

```
make
```

Alguns testes são conhecidos por travarem, ocasionalmente, indefinidamente. Portanto, para testar os resultados, execute a suíte de teste, mas configure um limite de tempo de dois minutos para cada caso de teste:

```
make test TESTOPTS="--timeout 120"
```

Para um sistema relativamente lento, você possivelmente precise aumentar o limite de tempo e 1 UPC (medido ao construir a passagem 1 do Binutils com um núcleo da CPU) deveria ser suficiente. Alguns testes são instáveis, de forma que a suíte de teste automaticamente reexecutará testes falhos. Se um teste falhou, mas então passou quando reexecutado, ele deveria ser considerado como passado.

Dois testes, `test_urllib2` e `test_urllibnet`, são conhecidos por falharem porque a resolução de nomes não está configurada no ambiente incompleto do LFS.

Instale o pacote:

```
make install
```

Nós usamos o comando **pip3** para instalar programas e módulos do Python 3 para todos(as) os(as) usantes como `root` em vários lugares neste livro. Isso conflita com a recomendação dos(as) desenvolvedores(as) do Python: instalar pacotes em um ambiente virtual ou no diretório lar de um(a) usante regular (executando **pip3** como esse(a) usante). Um aviso multi linhas será deflagrado sempre que **pip3** for emitido pelo(a) usante `root`.

A razão principal para a recomendação é para evitar conflitos com o gerenciador de pacote do sistema (**dpkg**, por exemplo). O LFS não tem um gerenciador de pacote abrangente ao sistema, de modo que isso não é um problema. Também, o **pip3** verificará se existe uma nova versão dele próprio sempre for executado. Uma vez que a resolução de nome de domínio ainda não está configurada no ambiente chroot do LFS, o **pip3** não consegue verificar se existe uma nova versão dele próprio e produzirá um aviso.

Depois que nós inicializarmos o sistema LFS e configurarmos uma conexão de rede de intercomunicação, um aviso diferente será emitido, informando para o(a) usante atualizar o **pip3** a partir de uma roda pré construída em PyPI (sempre que uma nova versão estiver disponível). Porém, o LFS considera que o **pip3** é uma parte do Python 3, de modo que ele não deveria ser atualizado separadamente. Além disso, uma atualização a partir de uma roda pré construída se desviaria do nosso objetivo: construir um sistema Linux a partir do código fonte. Assim, o aviso a respeito da nova versão do **pip3** deveria ser ignorado também. Se desejar, você pode suprimir todos esses avisos executando o seguinte comando, o qual cria um arquivo de configuração:

```
cat > /etc/pip.conf << EOF
[global]
root-user-action = ignore
disable-pip-version-check = true
EOF
```



Importante

No LFS e no BLFS normalmente nós construímos e instalamos módulos do Python com o comando **pip3**. Por favor, tenha certeza de que os comandos **pip3 install** em ambos os livros sejam executados como o(a) usante **root** (a menos que seja para um ambiente virtual do Python). Executar **pip3 install** como um(a) usante não **root** possivelmente aparente funcionar, porém causará o módulo instalado ficar inacessível por outros(as) usantes.

O **pip3 install** não reinstalará um módulo já instalado automaticamente. Quando usar o comando **pip3 install** para atualizar um módulo (por exemplo, de meson-0.61.3 para meson-0.62.0), insira a opção **--upgrade** na linha de comando. Se realmente for necessário desatualizar um módulo ou reinstalar a mesma versão por alguma razão, [então] insira **--force-reinstall --no-deps** na linha de comando.

Se desejado, então instale a documentação pré-formatada:

```
install -v -dm755 /usr/share/doc/python-3.14.7/html

tar --strip-components=1 \
  --no-same-owner \
  --no-same-permissions \
  -C /usr/share/doc/python-3.14.7/html \
  -xvf ../python-3.14.7-docs-html.tar.bz2
```

O significado dos comandos de instalação da documentação:

--no-same-owner e **--no-same-permissions**

Garanta que os arquivos instalados tenham a titularidade de propriedade e as permissões corretas. Sem essas opções, o tar instalará os arquivos de pacote com os valores do(a) criador(a) desenvolvedor(a).

8.54.2. Conteúdo do Python 3

Programas instalados:	idle3, pip3, pydoc3, python3 e python3-config
Biblioteca instalada:	libpython3.14.so e libpython3.so
Diretórios instalados:	/usr/include/python3.14, /usr/lib/python3 e /usr/share/doc/python-3.14.7

Descrições Curtas

idle3 é um script encapsulador que abre um editor GUI ciente do Python. Para esse script executar, você precisa ter instalado Tk antes do Python, de forma que o módulo Tkinter do Python seja construído.

pip3	O instalador de pacote para Python. Você pode usar pip para instalar pacotes originários do Python Package Index e outros índices.
pydoc3	é a ferramenta de documentação do Python
python3	é o interpretador para o Python, uma linguagem de programação orientada a objeto, interativa e interpretada

8.55. Flit-Core-4.0.2

Flit-core são as partes de construção de distribuição do Flit (uma ferramenta de empacotamento para módulos simples do Python).

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 1,3 MB

8.55.1. Instalação do Flit-Core

Construa o pacote:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instale o pacote:

```
pip3 install --no-index --find-links dist flit_core
```

O significado das opções e comandos de configuração do pip3:

wheel

Esse comando constrói o arquivamento wheel para este pacote.

-w dist

Instrui o pip a colocar a roda criada no diretório `dist`.

--no-cache-dir

Impede que o "pip" copie a roda criada para o diretório `/root/.cache/pip`.

install

Esse comando instala o pacote.

--no-build-isolation, *--no-deps* e *--no-index*

Essas opções impedem a busca de arquivos oriundos do repositório de pacotes online (PyPI). Se os pacotes fossem instalados na ordem correta, o pip não precisaria buscar quaisquer arquivos em primeiro lugar; essas opções adicionam alguma segurança em caso de erro do(a) usante.

--find-links dist

Instrui o pip a procurar por arquivamentos wheel no diretório `dist`.

8.55.2. Conteúdo do Flit-Core

Diretório instalado: `/usr/lib/python3.14/site-packages/flit_core` e `/usr/lib/python3.14/site-packages/flit_core-4.0.2.dist-info`

8.56. Packaging-26.3

O módulo de empacotamento é uma biblioteca Python que fornece utilitários que implementam as especificações de interoperabilidade que claramente têm um comportamento correto (PEP440) ou se beneficiam muito de ter uma implementação compartilhada (PEP425). Isso inclui utilitários para tratamento de versões, especificadores, marcadores, etiquetas e requisitos.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 3,0 MB

8.56.1. Instalação do Packaging

Compile packaging com o seguinte comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instale packaging com o seguinte comando:

```
pip3 install --no-index --find-links dist packaging
```

8.56.2. Conteúdo do Packaging

Diretórios instalados: /usr/lib/python3.14/site-packages/packaging e /usr/lib/python3.14/site-packages/packaging-26.3.dist-info

8.57. Wheel-0.48.0

Wheel é uma biblioteca do Python que é a implementação de referência do padrão de empacotamento de roda do Python.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 768 KB

8.57.1. Instalação da Wheel

Compile Wheel com o seguinte comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instale Wheel com o seguinte comando:

```
pip3 install --no-index --find-links dist wheel
```

8.57.2. Conteúdo do Wheel

Aplicativo instalado: wheel

Diretórios instalados: /usr/lib/python3.14/site-packages/wheel e /usr/lib/python3.14/site-packages/wheel-0.48.0.dist-info

Descrições Curtas

wheel é um utilitário para desempacotar, empacotar ou converter arquivamentos roda

8.58. Setuptools-84.0.0

"Setuptools" é uma ferramenta usada para baixar, construir, instalar, atualizar e desinstalar pacotes "Python".

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 19 MB

8.58.1. Instalação do Setuptools

Construa o pacote:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instale o pacote:

```
pip3 install --no-index --find-links dist setuptools
```

8.58.2. Conteúdo do Setuptools

Diretório instalado: /usr/lib/python3.14/site-packages/_distutils_hack, /usr/lib/python3.14/site-packages/pkg_resources, /usr/lib/python3.14/site-packages/setuptools e /usr/lib/python3.14/site-packages/setuptools-84.0.0.dist-info

8.59. Meson-1.12.0

Meson é um sistema de construção de fonte aberto projetado para ser ambos extremamente rápido e tão amigável para o(a) usante quanto possível.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 50 MB

8.59.1. Instalação do Meson

Compile Meson com o seguinte comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

A suíte de teste exige alguns pacotes fora do escopo do LFS.

Instale o pacote:

```
pip3 install --no-index --find-links dist meson
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/completions/meson
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_meson
```

O significado dos parâmetros do install:

-w dist

Coloca as rodas criadas no diretório dist.

--find-links dist

Instala as rodas a partir do diretório dist.

8.59.2. Conteúdo do Meson

Programas instalados: meson

Diretório instalado: /usr/lib/python3.14/site-packages/meson-1.12.0.dist-info e /usr/lib/python3.14/site-packages/mesonbuild

Descrições Curtas

meson Um sistema de construção de alta produtividade

8.60. Kmod-34.2

O pacote Kmod contém bibliotecas e utilitários para carregar módulos de núcleo

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 6,8 MB

8.60.1. Instalação do Kmod

Prepare Kmod para compilação:

```
mkdir -p build
cd      build

meson setup --prefix=/usr .. \
            --buildtype=release \
            -D manpages=false
```

O significado das opções do configure:

-D manpages=false

Essa opção desabilita gerar as páginas de manual, o que exige um programa externo.

Compile o pacote:

```
ninja
```

A suíte de teste desse pacote exige cabeçalhos crus de núcleo (não os cabeçalhos “sanitizados” de núcleo instalados anteriormente), os quais estão além do escopo do LFS.

Agora instale o pacote:

```
ninja install
```

8.60.2. Conteúdo do Kmod

Programas instalados: depmod (link para kmod), insmod (link para kmod), kmod, lsmod (link para kmod), modinfo (link para kmod), modprobe (link para kmod) e rmmod (link para kmod)

Biblioteca instalada: libkmod.so

Descrições Curtas

depmod	Cria um arquivo de dependência baseado nos símbolos que ele encontrar no conjunto existente de módulos; esse arquivo de dependência é usado pelo modprobe para carregar automaticamente os módulos exigidos
insmod	Instala um módulo carregável no núcleo em execução
kmod	Carrega e descarrega módulos de núcleo
lsmod	Lista módulos atualmente carregados
modinfo	Examina um arquivo objeto associado com um módulo de núcleo e exibe qualquer informação que ele consiga coletar
modprobe	Usa um arquivo de dependência, criado pelo depmod , para carregar automaticamente módulos relevantes
rmmod	Descarrega módulos a partir do núcleo em execução
libkmod	Essa biblioteca é usada por outros aplicativos para carregar e descarregar módulos de núcleo

8.61. Coreutils-9.11

O pacote Coreutils contém aplicativos utilitários básicos necessários por cada sistema operacional.

Tempo aproximado de construção: 1,2 UPC

Espaço em disco exigido: 194 MB

8.61.1. Instalação do Coreutils

POSIX exige que aplicativos originários do Coreutils reconheçam limites de carácter corretamente, mesmo em localidades multi bytes. O seguinte remendo corrige essa não-conformidade e outros defeitos relacionados à internacionalização.

```
patch -Np1 -i ../coreutils-9.11-i18n-1.patch
```



Nota

Muitos defeitos tem sido encontrados nesse remendo. Quando informar novos defeitos para os(as) mantenedores(as) do Coreutils, por favor, verifique primeiro para ver se tais defeitos são reproduzíveis sem esse remendo.

Agora prepare Coreutils para compilação:

```
autoreconf -fv
automake -af
FORCE_UNSAFE_CONFIGURE=1 ./configure \
    --prefix=/usr
```

O significado dos comandos e opções do configure:

autoreconf -fv

O remendo para internacionalização modificou o sistema de construção, portanto os arquivos de configuração precisam ser regenerados. Normalmente, nós usaríamos a opção `-i` para atualizar os arquivos auxiliares padrão, mas para esse pacote não funciona porque `configure.ac` especificou uma versão antiga do `gettext`.

automake -af

Os arquivos auxiliares do `automake` não foram atualizados pelo **autoconf** devido à ausência da opção `-i`. Esse comando os atualiza para evitar uma falha de construção.

```
FORCE_UNSAFE_CONFIGURE=1
```

Essa variável de ambiente permite que o pacote seja construído pelo(a) usante `root`.

Compile o pacote:

```
make
```

Pule para “Instale o pacote” se não executar a suíte de teste.

Agora a suíte de teste está pronta para ser executada. Primeiro, execute os testes que são destinados a serem executados como usante `root`:

```
make NON_ROOT_USERNAME=tester check-root
```

Nós vamos executar o resto dos testes como o(a) usante `tester`. Certos testes exigem que o(a) usante seja um(a) membro(a) de mais que um grupo. Portanto, para que esses testes não sejam ignorados, adicione um grupo temporário e torne o(a) usante `tester` uma parte dele:

```
groupadd -g 102 dummy -U tester
```

Corrija algumas das permissões, de modo que o(a) usante não root consiga compilar e executar os testes:

```
chown -R tester .
```

Agora execute os testes (usando `/dev/null` para a entrada padrão, ou dois testes possivelmente sejam quebrados se construir o LFS em um terminal gráfico ou uma sessão em SSH ou GNU Screen porque a entrada padrão é conectada a um PTY a partir da distribuição anfitriã, e o nó de dispositivo para tal PTY não pode ser acessado a partir do ambiente chroot do LFS):

```
su tester -c "PATH=$PATH make -k RUN_EXPENSIVE_TESTS=yes check" \
< /dev/null
```

Remova o grupo temporário:

```
groupdel dummy
```

Instale o pacote:

```
make install
```

Mova aplicativos para os locais especificados pelo FHS:

```
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"/' /usr/share/man/man8/chroot.8
```

8.61.2. Conteúdo do Coreutils

Programas instalados:	[, b2sum, base32, base64, basename, basenc, cat, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami e yes
Biblioteca instalada:	libstdbuf.so (em /usr/libexec/coreutils)
Diretório instalado:	/usr/libexec/coreutils

Descrições Curtas

[	É um comando atual, /usr/bin/[; é um sinônimo para o comando test
base32	Codifica e decodifica dados de acordo com a especificação base32 (RFC 4648)
base64	Codifica e decodifica dados de acordo com a especificação base64 (RFC 4648)
b2sum	Imprime ou verifica somas de verificação BLAKE2 (512 bits)
basename	Remove qualquer caminho e um dado sufixo de um nome de arquivo
basenc	Codifica ou decodifica dados usando vários algoritmos
cat	Concatena arquivos para saída gerada padrão
chgrp	Muda a propriedade do grupo de arquivos e diretórios
chmod	Muda as permissões de cada arquivo para o modo dado; o modo pode ser ou uma representação simbólica das mudanças a serem feitas ou um número octal representando as novas permissões
chown	Muda a titularidade de propriedade de usante e (ou) de grupo de arquivos e de diretórios
chroot	Executa um comando com o diretório especificado como o diretório /

cksum	Imprime a soma de verificação Cyclic Redundancy Check (CRC) e as contagens de bytes de cada arquivo especificado
comm	Compara dois arquivos ordenados, produzindo em três colunas as linhas que são únicas e as linhas que são comuns
cp	Copia arquivos
csplit	Divide um dado arquivo em vários novos arquivos, separando-os de acordo com padrões dados ou números de linha e produzindo a contagem de bytes de cada novo arquivo
cut	Imprime seções de linhas, selecionando as partes de acordo com campos ou posições dados
date	Exibe a data e hora atual no formato dado ou configura a data e hora do sistema
dd	Copia um arquivo usando o tamanho de bloco e a contagem dado, enquanto opcionalmente realiza conversões sobre ele
df	Informa a quantidade de espaço em disco disponível (e usada) em todos os sistemas de arquivos montados ou somente nos sistemas de arquivos contendo os arquivos selecionados
dir	Lista o conteúdo de cada diretório dado (o mesmo que o comando ls)
dircolors	Produz comandos para configurar a variável de ambiente LS_COLOR para mudar o esquema de cores usado por ls
dirname	Extraí a(s) porção(ões) de diretório(s) do(s) nome(s) dado(s)
du	Informa a quantidade de espaço em disco usado pelo diretório atual, por cada dos diretórios dados (incluindo todos os subdiretórios) ou por cada dos arquivos dados
echo	Exibe as sequências de caracteres dadas
env	Executa um comando em um ambiente modificado
expand	Converte tabulação para espaços
expr	Avalia expressões
factor	Imprime os fatores primos dos inteiros especificados
false	Não faz nada, sem sucesso; sempre sai com um código de status indicando falha
fmt	Reformata os parágrafos nos arquivos dados
fold	Quebra as linhas nos arquivos dados
groups	Informa as associações de grupo de um(a) usante
head	Imprime as primeiras dez linhas (ou o número de linhas dado) de cada arquivo dado
hostid	Informa o identificador numérico (em hexadecimal) do dispositivo
id	Informa o efetivo ID de usante, ID de grupo e as associações de grupo do(a) usante atual ou usante especificado(a)
install	Copia arquivos enquanto configura os modos de permissão deles e, se possível, proprietário e grupo deles
join	Junta as linhas que tenham idênticos campos de junção a partir de dois arquivos
link	Cria um link rígido (com o nome dado) para um arquivo
ln	Faz links rígidos ou links flexíveis (simbólicos) entre arquivos
logname	Informa o nome de login do(a) usante atual
ls	Lista o conteúdo de cada diretório dado
md5sum	Informa ou verifica somas de verificação Message Digest 5 (MD5)
mkdir	Cria diretórios com os nomes dados

mkfifo	Cria First-In, First-Outs (FIFOs), "tubos nomeado" na linguagem UNIX, com os nomes dados
mknod	Cria nós de dispositivo com os nomes dados; um nó de dispositivo é um arquivo especial de caractere, um arquivo especial de bloco ou um FIFO
mktemp	Cria arquivos temporários de uma maneira segura; é usado em conjuntos de comandos sequenciais
mv	Move ou renomeia arquivos ou diretórios
nice	Executa um aplicativo com prioridade de agendamento modificada
nl	Numera as linhas a partir dos arquivos dados
nohup	Executa um comando imune a interrupções, com a saída gerada dele redirecionada para um arquivo de registro
nproc	Imprime o número de unidades de processamento disponíveis para um processo
numfmt	Converte números para ou oriundos de sequências de caracteres legíveis por humanos
od	Despeja arquivos em octal e outros formatos
paste	Mescla os arquivos dados, unindo linhas sequencialmente correspondentes lado a lado, separadas por caracteres de tabulação
pathchk	Verifica se nomes de arquivos são válidos ou portáteis
pinky	É um cliente leve e apontador; ele informa algumas informações a respeito dos(as) usantes fornecidos(as)
pr	Pagina e coluna arquivos para impressão
printenv	Imprime o ambiente
printf	Imprime os argumentos dados de acordo com o formato dado, muito parecido com a função printf do C
ptx	Produz um índice permutado a partir do conteúdo dos arquivos dados, com cada palavra-chave no contexto dela
pwd	Informa o nome do diretório de trabalho atual
readlink	Informa o valor do link simbólico dado
realpath	Imprime o caminho resolvido
rm	Remove arquivos ou diretórios
rmdir	Remove diretórios se eles estiverem vazios
seq	Imprime uma sequência de números dentro de um dado intervalo e com um dado incremento
sha1sum	Imprime ou verifica somas de verificação do Secure Hash Algorithm 1 (SHA1) 160 bits
sha224sum	Imprime ou verifica somas de verificação do Secure Hash Algorithm de 224 bits
sha256sum	Imprime ou verifica somas de verificação do Secure Hash Algorithm de 256 bits
sha384sum	Imprime ou verifica somas de verificação do Secure Hash Algorithm de 384 bits
sha512sum	Imprime ou verifica somas de verificação do Secure Hash Algorithm de 512 bits
shred	Sobrescreve os arquivos dados repetidamente com padrões complexos, tornando difícil recuperar os dados
shuf	Embaralha linhas do texto
sleep	Pausa pelo período de tempo dado
sort	Ordena as linhas a partir dos arquivos dados
split	Divide o arquivo dado em pedaços, por tamanho ou por número de linhas

stat	Exibe a situação de arquivo ou sistema de arquivos
stdbuf	Executa comandos com operações de buffer alteradas para fluxos padrão deles
stty	Configura ou informa configurações de linha de terminal
sum	Imprime soma de verificação e contagens de blocos para cada arquivo dado
sync	Libera buffers do sistema de arquivos; isso força blocos modificados para o disco e atualiza o super bloco
tac	Concatena os arquivos dados em ordem reversa
tail	Imprime as últimas dez linhas (ou o número dado de linhas) de cada arquivo dado
tee	Lê a partir da entrada gerada padrão enquanto escreve tanto para a saída gerada padrão quanto para os arquivos dados
test	Compara valores e verifica tipos de arquivos
timeout	Executa um comando com um limite de tempo
touch	Muda carimbos de tempo de arquivo, definindo os horários de acesso e modificação dos arquivos dados para o horário atual; arquivos que não existem são criados com tamanho zero
tr	Traduz, comprime e deleta os caracteres dados a partir da entrada gerada padrão
true	Não faz nada, com sucesso; sempre sai com um código de situação indicando sucesso
truncate	Reduz ou expande um arquivo para o tamanho especificado
tsort	Realiza uma ordenação topológica; ele escreve uma lista completamente ordenada de acordo com a ordenação parcial em um arquivo dado
tty	Informa o nome de arquivo do terminal conectado à entrada gerada padrão
uname	Informa informação de sistema
unexpand	Converte espaços para tabulação
uniq	Descarta todas, exceto uma das sucessivas linhas idênticas
unlink	Remove o arquivo dado
users	Informa os nomes dos(as) usantes atualmente logados(as)
vdir	É o mesmo que ls -l
wc	Informa o número de linhas, palavras e bytes para cada arquivo dado, bem como totais gerais quando mais que um arquivo for dado
who	Informa quem está logado(a)
whoami	Informa o nome de usante associado com o ID efetivo atual de usante
yes	Repetidamente produz y ou uma dada sequência de caracteres, até ser eliminado
libstdbuf	Biblioteca usada por stdbuf

8.62. Diffutils-3.12

O pacote Diffutils contém aplicativos que mostram as diferenças entre arquivos ou diretórios.

Tempo aproximado de 0,4 UPC

construção:

Espaço em disco exigido: 51 MB

8.62.1. Instalação do Diffutils

Prepare o Diffutils para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.62.2. Conteúdo do Diffutils

Programas instalados: cmp, diff, diff3 e sdiff

Descrições Curtas

cmp	Compara dois arquivos e informa quaisquer diferenças bytes por bytes
diff	Compara dois arquivos ou diretórios e informa quais linhas nos arquivos diferem
diff3	Compara três arquivos linha por linha
sdiff	Mescla dois arquivos e interativamente exibe os resultados

8.63. Findutils-4.11.0

O pacote Findutils contém aplicativos para encontrar arquivos. Os aplicativos são fornecidos para procurar ao longo de todos os arquivos em uma árvore de diretórios e para criar, manter e buscar uma base de dados (geralmente mais rápido que o find recursivo, porém não é confiável, a menos que a base de dados tenha sido atualizada recentemente). O Findutils também abastece o aplicativo **xargs**, o qual pode ser usado para executar um comando especificado sobre cada arquivo selecionado por uma pesquisa.

Tempo aproximado de 0,9 UPC

construção:

Espaço em disco exigido: 71 MB

8.63.1. Instalação do Findutils

Prepare o Findutils para compilação:

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

O significado das opções do configure:

--localstatedir

Essa opção move a base de dados **locate** para /var/lib/locate, o qual é o local conforme com FHS.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
chown -R tester .
su tester -c "PATH=$PATH make check -k"
```

Um teste nominado test-regex-e1 é conhecido por falhar.

Instale o pacote:

```
make install
```

8.63.2. Conteúdo do Findutils

Programas instalados: find, locate, updatedb e xargs

Diretório instalado: /var/lib/locate

Descrições Curtas

find	Pesquisa nas árvores de diretórios dadas por arquivos correspondendo a critérios especificados
locate	Pesquisa em uma base de dados de nomes de arquivo e informa os nomes que contém uma sequência de caracteres dada ou correspondem a um padrão dado
updatedb	Atualiza a base de dados locate ; ele escaneia o sistema de arquivos inteiro (incluindo outros sistemas de arquivos que estejam montados atualmente, a menos que dito o contrário) e coloca cada nome de arquivo que ele encontrar na base de dados
xargs	Pode ser usado para aplicar um comando dado a uma lista de arquivos

8.64. Groff-1.24.1

O pacote Groff contém aplicativos para processar e formatar texto e imagens.

Tempo aproximado de construção: 0,6 UPC

Espaço em disco exigido: 128 MB

8.64.1. Instalação do Groff

Groff espera que a variável de ambiente `PAGE` contenha o tamanho padrão de papel. Para usantes nos Estados Unidos da América do Norte, `PAGE=letter` é apropriado. Em outros lugares, `PAGE=A4` possivelmente seja mais adequado. Embora o tamanho padrão do papel seja configurado durante a compilação, ele pode ser substituído posteriormente ecoando-se ou “A4” ou “letter” para o arquivo `/etc/papersize`.

Prepare Groff para compilação:

```
PAGE=<tamanho_papel> ./configure --prefix=/usr
```

Construa o pacote:

```
make -j1
```

Para testar os resultados, emita:

```
make check
```

Um teste, `neqn-smoke-test.sh`, é conhecido por falhar.

Instale o pacote:

```
make install
```

8.64.2. Conteúdo do Groff

Programas instalados: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropsdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit e troff

Diretórios instalados: /usr/lib/groff, /usr/share/doc/groff-1.24.1 e /usr/share/groff

Descrições Curtas

addftinfo	Lê um arquivo de fonte troff e acrescenta alguma informação de métrica de fonte adicional que é usada pelo sistema groff
afmtodit	Cria um arquivo de fonte para uso com groff e grops
chem	Preprocessador Groff para produzir diagramas de estrutura química
eqn	Compila descrições de equações embutidas em arquivos de entrada gerada do troff em comandos que são entendidos pelo troff
eqn2graph	Converte uma EQN (equação) do troff em uma imagem recortada
gdiffmk	Marca diferenças entre arquivos groff/nroff/troff
glilypond	Transforma partituras escritas na linguagem lilypond na linguagem groff
gperl	Preprocessador para groff, permitindo a inserção de código perl em arquivos groff
gpinyin	Preprocessador para groff, permitindo a inserção do Pinyin (Chinês Mandarim escrito com o alfabeto romano) em arquivos groff.

grap2graph	Converte um arquivo do aplicativo grap em uma imagem recortada bitmap (grap é uma linguagem de programação antiga do Unix para criar diagramas)
grn	Um pré-processador groff para arquivos gremlin
grodvi	Um controlador para groff que produz arquivos de saída gerada do formato dvi do TeX
groff	Uma estrutura de interação direta com o(a) usuário para o sistema de formatação de documentos groff ; ele executa o programa troff e um pós-processador apropriado para o dispositivo selecionado
groffer	Exibe arquivos groff e páginas de manual em terminais X e tty
grog	Lê arquivos e advinha quais das opções do groff -e, -man, -me, -mm, -ms, -p, -s e -t são exigidas para imprimir arquivos e informa ao comando groff incluindo aquelas opções
grolbp	É um controlador groff para impressoras Canon CAPSL (impressoras a laser séries LBP-4 e LBP-8)
grolj4	É um controlador para groff que produz saída gerada no formato PCL5 adequado para uma impressora HP LaserJet 4
gropdf	Traduz a saída gerada do GNU troff para PDF
grops	Traduz a saída gerada do GNU troff para PostScript
grotty	Traduz a saída gerada do GNU troff em uma forma adequada para dispositivos semelhantes a máquina de escrever
hpftodit	Cria um arquivo de fonte para uso com groff -Tlj4 a partir de um arquivo de métrica de fonte rotulada HP
indxbib	Cria um índice invertido para as bases de dados bibliográficas com um arquivo especificado para uso com refer , lookbib e lkbib
lkbib	Pesquisa em bases de dados bibliográficas por referências que contenham chaves especificadas e informa quaisquer referências encontradas
lookbib	Imprime um prompt na saída de erro padrão (a menos que a entrada gerada padrão não seja um terminal); lê uma linha contendo um conjunto de palavras chave a partir da entrada gerada padrão; pesquisa nas bases de dados bibliográficas, em um arquivo especificado, por referências contendo aquelas palavras chave; imprime quaisquer referências encontradas na saída gerada padrão; e repete esse processo até o final da entrada gerada
mmroff	Um pré-processador simples para groff
neqn	Formata equações para saída gerada American Standard Code for Information Interchange (ASCII)
nroff	Um script que emula o comando nroff usando groff
pdfmom	É um encapsulador em torno do groff que facilita a produção de documentos PDF a partir de arquivos formatados com as macros mom .
pdfroff	Cria documentos pdf usando groff
pfbtops	Traduz uma fonte PostScript em formato .pfb para ASCII
pic	Compila descrições de imagens embutidas em arquivos de entrada gerada troff ou TeX em comandos entendidos pelo TeX ou troff
pic2graph	Converte um diagrama PIC em uma imagem recortada
post-grohtml	Traduz a saída gerada do GNU troff para HTML
preconv	Converte codificação de arquivos de entrada gerada para alguma coisa que o GNU troff entenda
pre-grohtml	Traduz a saída gerada do GNU troff para HTML

refer	Copia o conteúdo de um arquivo para a saída gerada padrão, exceto aquelas linhas entre <i>./</i> e <i>./</i> que são interpretadas como citações e linhas entre <i>.R1</i> e <i>.R2</i> que são interpretadas como comandos para como citações são para serem processadas
roff2dvi	Transforma arquivos roff para o formato DVI
roff2html	Transforma arquivos roff para o formato HTML
roff2pdf	Transforma arquivos roff em PDFs
roff2ps	Transforma arquivos roff para arquivos ps
roff2text	Transforma arquivos roff para arquivos de texto
roff2x	Transforma arquivos roff para outros formatos
soelim	Lê arquivos e substitui linhas da forma <i>.so arquivo</i> pelo conteúdo do <i>arquivo</i> mencionado
tbl	Compila descrições de tabelas embutidas em arquivos de entrada gerada do troff em comandos que são entendidos pelo troff
tfmtodit	Cria um arquivo fonte para uso com groff -Tdvi
troff	É altamente compatível com o troff do Unix; ele usualmente deveria ser invocado usando o comando groff , o qual também executará preprocessadores e pós-processadores na ordem apropriada e com as opções apropriadas

8.65. GRUB-2.14

O pacote GRUB contém o GRand Unified Bootloader.



Nota

Esta página está dividida em múltiplas seções destinadas para instalar para um método específico de inicialização (BIOS, UEFI de 64 bits e UEFI de 32 bits). O GRUB não pode ser construído com todas as arquiteturas de métodos de inicialização de uma vez.

Você pode ignorar outras seções para ir para o método de inicialização que precisa. Se em dúvida, você pode seguir todas as seções, ao custo de tempo extra de construção. Depois que você tiver instalado suporte para teu método de inicialização, continue construindo o restante dos pacotes neste capítulo. Tornar teu sistema LFS inicializável com GRUB será discutido na Seção 10.4, “Usando o GRUB para Configurar o Processo de Inicialização.”



Atenção

Desconfigure quaisquer variáveis de ambiente que possivelmente afetem a construção:

```
unset {C,CPP,CXX,LD}FLAGS
```

Não tente “ajustar” esse pacote com sinalizadores personalizados de compilação. Esse pacote é um carregador de inicialização. As operações de baixo nível no código fonte possivelmente sejam quebradas por otimização agressiva.

Tempo aproximado de construção: 1,0 UPC
Espaço em disco exigido: 245 MB

8.65.1. Instalação do GRUB para BIOS

Primeiro corrija um defeito introduzido no grub-2.14:

```
sed 's/--image-base/--nonexist-linker-option/' -i configure
```

Prepare GRUB para compilação:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --disable-efiemu \
            --disable-werror
```

O significado das novas opções de configurar:

`--disable-werror`

Isso permite que a construção complete com avisos introduzidos por versões mais recentes do Flex.

`--disable-efiemu`

Essa opção minimiza o que é construído desabilitando um recurso e eliminando alguns programas de teste não necessários para o LFS.

Compile o pacote:

```
make
```

A suíte de teste para esse pacote não é recomendada. A maioria dos testes depende de pacotes que não estão disponíveis no ambiente limitado do LFS. Para executar os testes mesmo assim, execute **make check**.

Instale o pacote:

```
make install
```

8.65.2. Instalação do GRUB para UEFI de 64 Bits

Se quiser inicializar com UEFI de 64 bits, você deveria construir suporte para ela.

Primeiro, se você construiu o GRUB a partir da seção acima, limpe a árvore do fonte:

```
make clean
```

Agora configure o GRUB para suporte UEFI de 64 bits:

```
./configure --prefix=/usr      \
            --sysconfdir=/etc   \
            --target=x86_64     \
            --with-platform=efi \
            --disable-efiemu    \
            --disable-werror
```

O significado das novas opções de configurar:

```
--target=x86_64
```

Isso define que a arquitetura do firmware UEFI é x86_64, a qual o GRUB deveria alvejar.

```
--with-platform=efi
```

Isso especifica que EFI é uma plataforma que o GRUB deveria alvejar. Em combinação com `--target=x86_64`, o GRUB terá a capacidade para alvejar a plataforma x86_64-efi.

Compile o pacote para suporte UEFI de 64 bits:

```
make
```

Instale suporte para UEFI de 64 bits:

```
make install
```

8.65.3. Instalação do GRUB para UEFI de 32 Bits

Se quiser inicializar com UEFI de 32 bits, o que é muito raro, você deveria construir suporte para ela.

Primeiro, se você construiu o GRUB a partir de alguma das seções acima, limpe a árvore do fonte:

```
make clean
```

Agora configure o GRUB para suporte UEFI de 32 bits:

```
./configure --prefix=/usr      \
            --sysconfdir=/etc   \
            --target=i386       \
            --with-platform=efi \
            --disable-efiemu    \
            --disable-werror
```

O significado das novas opções de configurar:

```
--target=i386
```

Isso define que a arquitetura de firmware UEFI é i386/32 bits, que o GRUB deveria alvejar. Em combinação com `--with-platform=efi`, o GRUB terá a capacidade de alvejar a plataforma i386-efi.

Compile o pacote para suporte UEFI de 32 bits:

```
make
```

Instale o suporte para UEFI de 32 bits:

```
make install
```

8.65.4. Conteúdo do GRUB

Programas instalados:	grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimimage, 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 e grub-syslinux2cfg
Diretórios instalados:	/usr/lib/grub, /etc/grub.d, /usr/share/grub e /boot/grub (quando grub-install for primeiro executado)



Nota

/usr/lib/grub terá conteúdo diferente baseado em qual(is) plataforma(s) você tiver instalado o GRUB. Ou seja, existirão diferentes módulos do GRUB para cada plataforma.

Descrições Curtas

grub-bios-setup	É um programa auxiliar para grub-install
grub-editenv	Uma ferramenta para editar o bloco ambiente
grub-file	Verifica para ver se o arquivo fornecido é do tipo especificado
grub-fstest	É uma ferramenta para depurar o controlador do sistema de arquivos
grub-glue-efi	Cola binários de 32 bits e de 64 bits em um arquivo (para máquinas Apple)
grub-install	Instala o GRUB na tua unidade
grub-kbdcomp	É um conjunto de comandos sequenciais que converte um esquema xkb em um reconhecido pelo GRUB
grub-macbless	É o bless ao estilo Mac para sistemas de arquivos HFS ou HFS+ (bless é peculiar às máquinas Apple; ele torna um dispositivo inicializável)
grub-menulst2cfg	Converte um menu.lst do GRUB Legado em um grub.cfg para uso com GRUB 2
grub-mkconfig	Gera um arquivo grub.cfg
grub-mkimimage	Faz uma imagem inicializável do GRUB
grub-mklayout	Gera um arquivo de esquema de teclado do GRUB
grub-mknetdir	Prepara um diretório do GRUB de inicialização de rede de comunicação
grub-mkpasswd-pbkdf2	Gera uma senha encriptada PBKDF2 para uso no menu de inicialização
grub-mkrelpath	Torna um nome de caminho de sistema relativo à raiz dele
grub-mkrescue	Faz uma imagem inicializável do GRUB adequada para um disquete, CDROM/DVD ou uma unidade USB
grub-mkstandalone	Gera uma imagem independente
grub-ofpathname	É um programa auxiliar que imprime o caminho para um dispositivo do GRUB
grub-probe	Sonda informação de dispositivo para um caminho ou dispositivo fornecido
grub-reboot	Configura a entrada padrão de inicialização para o GRUB somente para a próxima inicialização
grub-render-label	Renderiza .disk_label da Apple para Macs da Apple

grub-script-check	Verifica conjunto de comandos sequenciais de configuração do GRUB para erros de sintaxe
grub-set-default	Configura a entrada padrão de inicialização para o GRUB
grub-sparc64-setup	É um programa auxiliar para grub-setup
grub-syslinux2cfg	Transforma um arquivo de configuração do syslinux para o formato grub.cfg

8.66. Gzip-1.14

O pacote Gzip contém aplicativos para comprimir e descomprimir arquivos.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 21 MB

8.66.1. Instalação do Gzip

Prepare o Gzip para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.66.2. Conteúdo do Gzip

Programas instalados: gunzip, gzexe, gzip, uncompress (link rígido com gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore e znew

Descrições Curtas

gunzip	Descomprime arquivos gzipados
gzexe	Cria arquivos executáveis auto-descomprimíveis
gzip	Comprime os arquivos dados usando codificação Lempel-Ziv (LZ77)
uncompress	Descomprime arquivos comprimidos
zcat	Descomprime os arquivos gzipados dados para a saída gerada padrão
zcmp	Executa cmp em arquivos gzipados
zdiff	Executa diff em arquivos gzipados
zegrep	Executa egrep em arquivos gzipados
zfgrep	Executa fgrep em arquivos gzipados
zforce	Força uma extensão .gz sobre todos os arquivos dados que sejam arquivos gzipados, de modo que o gzip não os comprimirá novamente; isso pode ser útil quando os nomes de arquivo foram truncados durante uma transferência de arquivo
zgrep	Executa grep em arquivos gzipados
zless	Executa less em arquivos gzipados
zmore	Executa more em arquivos gzipados
znew	Re-comprime arquivos oriundos do formato compress para o formato gzip —.z para .gz

8.67. IPRoute2-7.1.0

O pacote IPRoute2 contém aplicativos para operação interativa básica e avançada de dispositivos via rede de comunicação baseada em IPV4.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 18 MB

8.67.1. Instalação do IPRoute2

O aplicativo **arpd** incluído nesse pacote não será construído, dado que ele é dependente do Berkeley DB, o qual não é instalado no LFS. Entretanto, um diretório e uma página de manual para o **arpd** ainda serão instalados. Impeça isso executando os comandos mostrados abaixo.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Compile o pacote:

```
make NETNS_RUN_DIR=/run/netns
```

Esse pacote não tem uma suíte de teste funcional.

Instale o pacote:

```
make SBINDIR=/usr/sbin install
```

Se desejado, instale a documentação:

```
install -vDm644 COPYING README* -t /usr/share/doc/iproute2-7.1.0
```

8.67.2. Conteúdo do IPRoute2

Programas instalados: bridge, ctstat (link para lstat), genl, ifstat, ip, lstat, nstat, routel, rtacct, rtmon, rtpr, rtstat (link para lstat), ss e tc

Diretórios instalados: /etc/iproute2, /usr/lib/tc e /usr/share/doc/iproute2-7.1.0

Descrições Curtas

bridge Configura pontes de redes de comunicação

ctstat Utilitário de situação de conexão

genl Estrutura genérica de interação direta com o(a) usante do utilitário de linque de rede de intercomunicação

ifstat Mostra as estatísticas de interface, incluindo o número de pacotes transmitidos e recebidos, por interface

ip O executável principal. Ele tem várias funções, incluindo estas:

ip link <dispositivo> permite que usantes olhem para o estado de dispositivos e façam mudanças

ip addr permite que usantes olhem para endereços e propriedades deles, adicionem novos endereços e deletem os antigos

ip neighbor permite que usantes olhem para vínculos de vizinho e propriedades deles, adicionem novas entradas de vizinho e deletem as antigas

ip rule permite que usantes olhem para as políticas de roteamento e as mudem

ip route permite que usantes olhem para a tabela de roteamento e mudem regras da tabela de roteamento

ip tunnel permite que usantes olhem para os tuneis IP e propriedades deles e as mudem

ip maddr permite que usantes olhem para os endereços multicast e propriedades deles e as mudem

ip mroute permite que usantes configurem, mudem ou deletem o roteamento multicast

	ip monitor permite que usantes continuamente monitorem o estado de dispositivos, endereços e rotas
lnstat	Fornece estatísticas de rede de comunicação do Linux; ele é uma substituição difundida e mais completa de recursos para o antigo aplicativo rtstat
nstat	Mostra estatísticas da rede de comunicação
routel	Um componente do ip route para listar as tabelas de roteamento
rtacct	Exibe o conteúdo de <code>/proc/net/route</code>
rtmon	Utilitário de monitoramento de rota
rtpr	Converte a saída gerada de ip -o em um formato legível
rtstat	Utilitário de situação de rota
ss	Similar ao comando netstat ; exibe conexões ativas
tc	Controle de tráfego para implementações de Quality Of Service (QoS) e Class Of Service (CoS)
	tc qdisc permite que usantes configurem a disciplina de enfileiramento
	tc class permite que usantes configurem classes baseadas no agendamento da disciplina de enfileiramento
	tc filter permite que usantes configurem a filtragem de pacote QoS/CoS
	tc monitor pode ser usado para visualizar mudanças feitas para o Traffic Control no núcleo.

8.68. Kbd-2.10.0

O pacote Kbd contém arquivos de tabelas de teclas, fontes de console e utilitários de teclado.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 45 MB

8.68.1. Instalação do Kbd

O comportamento das teclas backspace e delete não é consistente ao longo dos mapas de teclas no pacote Kbd. O seguinte remendo corrige esse problema para mapas de tecla i386:

```
patch -Np1 -i ../kbd-2.10.0-backspace-1.patch
```

Após remendar, a tecla backspace gera o carácter com código 127 e a tecla delete gera uma sequência de escape bem conhecida.

Remova o aplicativo redundante **resizecons** (ele exige que a defunta svgalib forneça os arquivos de modo de vídeo - para uso normal **setfont** dimensiona o console adequadamente) juntamente com a página de manual dele.

```
sed -i '/RESIZECONS_PROGS=/s/yes/no/' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Prepare Kbd para compilação:

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

O significado da opção de configure:

--disable-vlock

Essa opção evita que o utilitário vlock seja construído, pois ele exige a biblioteca PAM, a qual não está disponível no ambiente chroot.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```



Nota

Para alguns idiomas (por exemplo, Bielorrusso) o pacote Kbd não fornece um mapa de tecla útil onde o mapa de tecla “by” regular supõe a codificação ISO-8859-5 e o mapa de tecla CP1251 normalmente é usado. Usantes de tais idiomas tem que transferir mapas de tecla funcionais separadamente.

Se desejado, instale a documentação:

```
cp -R -v docs/doc -T /usr/share/doc/kbd-2.10.0
```

8.68.2. Conteúdo do Kbd

Programas instalados:	chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link para psfxtable), psfgettable (link para psfxtable), psfstriptime (link para psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start e unicode_stop
Diretórios instalados:	/usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.10.0 e /usr/share/unimaps

Descrições Curtas

chvt	Muda o terminal virtual de primeiro plano
deallocvt	Desaloca terminais virtuais não usados
dumpkeys	Despeja as tabelas de tradução de teclado
fgconsole	Imprime o número do terminal virtual ativo
getkeycodes	Imprime a tabela de mapeamento de código de escaneamento para código de tecla do núcleo
kbinfo	Obtém informação a respeito da situação de um console
kbd_mode	Informa ou configura o modo de teclado
kbdrate	Configura as taxas de repetição e de atraso do teclado
loadkeys	Carrega as tabelas de tradução do teclado
loadunimap	Carrega a tabela de mapeamento Unicode para fonte do núcleo
mapscrn	Um programa obsoleto que costumava carregar uma tabela de mapeamento de caractere de saída gerada definida pelo(a) usante para dentro do controlador de console; isso agora é feito por setfont
openvt	Inicia um aplicativo em um novo terminal virtual (VT)
psfaddtable	Adiciona uma tabela de carácter Unicode para uma fonte de console
psfgettable	Extrai a tabela de carácter Unicode embutida a partir de uma fonte de console
psfstriptime	Remove a tabela de carácter Unicode embutida a partir de uma fonte de console
psfxtable	Lida com tabelas de carácter Unicode para fontes de console
setfont	Muda as fontes Enhanced Graphic Adapter (EGA) e Video Graphics Array (VGA) no console
setkeycodes	Carrega entradas de tabela de mapeamento de código de escaneamento para código de tecla do núcleo; isso é útil se existirem teclas incomuns no teclado
setleds	Configura os sinalizadores de teclado e Light Emitting Diodes (LEDs)
setmetamode	Define o manuseio de meta tecla do teclado
setvtrgb	Configura o mapa de cor do console em todos os terminais virtuais
showconsolefont	Exibe a fonte de tela atual do console EGA/VGA
showkey	Informa os códigos de escaneamento, códigos de tecla e códigos ASCII das teclas pressionadas no teclado
unicode_start	Põe o teclado e o console em modo UNICODE [Não use esse programa, a menos que teu arquivo de mapa de teclas esteja na codificação ISO-8859-1. Para outras codificações, esse utilitário produz resultados incorretos].

unicode_stop

Reverte teclado e console do modo UNICODE

8.69. Libpipeline-1.5.8

O pacote Libpipeline contém uma biblioteca para manipular pipelines de subprocessos de uma maneira flexível e conveniente.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 10 MB

8.69.1. Instalação do Libpipeline

Prepare Libpipeline para compilação:

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

Compile o pacote:

```
make
```

Os testes exigem a biblioteca Check que nós removemos do LFS.

Instale o pacote:

```
make install
```

8.69.2. Conteúdo do Libpipeline

Biblioteca instalada: libpipeline.so

Descrições Curtas

libpipeline	Essa biblioteca é usada para construir seguramente pipelines entre subprocessos
-------------	---

8.70. Make-4.4.1

O pacote Make contém um aplicativo para controlar a geração de executáveis e outros arquivos não fonte de um pacote a partir de arquivos fonte.

Tempo aproximado de construção: 0,6 UPC

Espaço em disco exigido: 13 MB

8.70.1. Instalação do Make

Prepare o Make para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
chown -R tester .  
su tester -c "PATH=$PATH make check"
```

Instale o pacote:

```
make install
```

8.70.2. Conteúdo do Make

Aplicativo instalado: make

Descrições Curtas

make Automaticamente determina quais pedaços de um pacote precisam ser (re)compiladas e então emite os comandos relevantes

8.71. Patch-2.8

O pacote Patch contém um aplicativo para modificar ou criar arquivos por aplicação de um arquivo “remendo” tipicamente criado pelo aplicativo **diff**.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 14 MB

8.71.1. Instalação do Patch

Prepare o Patch para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.71.2. Conteúdo do Patch

Aplicativo instalado: patch

Descrições Curtas

patch Modifica arquivos de acordo com um arquivo de remendo (Um arquivo de remendo normalmente é uma listagem de diferenças criada com o programa **diff**. Aplicando-se essas diferenças aos arquivos originais, **patch** cria as versões remendadas).

8.72. Tar-1.35

O pacote Tar fornece a habilidade para criar arquivamentos tar bem como para realizar vários outros tipos de manipulação de arquivamento. Tar pode ser usado sobre arquivamentos previamente criados para extrair arquivos, para armazenar arquivos adicionais ou para atualizar ou listar arquivos que já foram armazenados.

Tempo aproximado de construção: 0,5 UPC

Espaço em disco exigido: 43 MB

8.72.1. Instalação do Tar

Primeiro corrija o pacote para construir com acl-2.4.0:

```
patch -Np1 -i ../tar-1.35-acl_fix-1.patch
```

Prepare o Tar para compilação:

```
FORCE_UNSAFE_CONFIGURE=1 \
./configure --prefix=/usr
```

O significado da opção de configure:

FORCE_UNSAFE_CONFIGURE=1

Isso força o teste para mknod ser executado como root. Geralmente é considerado perigoso executar esse teste como o(a) usante root, porém, como ele está sendo executado em um sistema que foi construído somente parcialmente, substituí-lo está OK.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Um teste, "capabilities: binary store/restore", é conhecido por falhar se for executado, pois o LFS carece de "selinux"; porém, será pulado se o núcleo do anfitrião não suportar atributos estendidos ou rótulos de segurança no sistema de arquivos usado para construir o LFS.

Instale o pacote:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.35
```

8.72.2. Conteúdo do Tar

Programas instalados: tar

Diretório instalado: /usr/share/doc/tar-1.35

Descrições Curtas

tar Cria, extrai arquivos originários de, e lista o conteúdo de, arquivamentos, também conhecidos como tarballs

8.73. Texinfo-7.3

O pacote Texinfo contém aplicativos para leitura, escrita e conversão de páginas info.

Tempo aproximado de construção: 0,4 UPC

Espaço em disco exigido: 151 MB

8.73.1. Instalação do Texinfo

Prepare o Texinfo para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

Opcionalmente, instale os componentes pertencentes a uma instalação do TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

O significado do parâmetro do make:

```
TEXMF=/usr/share/texmf
```

A variável de arquivo make TEXMF mantém o local da raiz da árvore do TeX se, por exemplo, um pacote do TeX será instalado posteriormente.

O sistema de documentação Info usa um arquivo de texto plano para manter a lista de entradas de menu dele. O arquivo está localizado em `/usr/share/info/dir`. Infelizmente, devido a problemas ocasionais nos arquivos Make de vários pacotes, ele pode às vezes sair de sincronia com as páginas info instaladas no sistema. Se o arquivo `/usr/share/info/dir` alguma vez precisar ser recriado, [então] os seguintes comandos opcionais realizarão a tarefa:

```
pushd /usr/share/info
rm -v dir
for f in *
do install-info $f dir 2>/dev/null
done
popd
```

8.73.2. Conteúdo do Texinfo

Programas instalados: info, install-info, makeinfo (link para texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf e texindex

Biblioteca instalada: MiscXS.so, Parsetexi.so e XSParagraph.so (todas em `/usr/lib/texinfo`)

Diretórios instalados: `/usr/share/texinfo` e `/usr/lib/texinfo`

Descrições Curtas

info Usado para ler páginas info as quais são similares a páginas de manual, porém frequentemente vão muito mais fundo que somente explicar todas as opções de linha de comando disponíveis [Por exemplo, compare **man bison** e **info bison**].

install-info	Usado para instalar páginas info; ele atualiza entradas no arquivo de índice info
makeinfo	Traduz os documentos fonte do Texinfo dados para páginas info, texto plano ou HTML
pdftexi2dvi	Usado para formatar o documento do Texinfo dado em um arquivo Portable Document Format (PDF)
pod2texi	Converte Pod para formato Texinfo
texi2any	Traduz documentação fonte do Texinfo para vários outros formatos
texi2dvi	Usado para formatar o documento do Texinfo dado em um arquivo independente de dispositivo que pode ser impresso
texi2pdf	Usado para formatar o documento do Texinfo dado em um arquivo Portable Document Format (PDF)
texindex	Usado para ordenar arquivos de índice do Texinfo

8.74. Vim-9.2.1025

O pacote Vim contém um editor poderoso de texto.

Tempo aproximado de 3,3 UPC

construção:

Espaço em disco exigido: 234 MB



Alternativas ao Vim

Se você preferir outro editor—como o Emacs, Joe ou Nano—por favor, consulte <https://www.linuxfromscratch.org/blfs/view/13.1-systemd/postlfs/editors.html> para instruções sugeridas de instalação.

8.74.1. Instalação do Vim

Primeiro, mude o local padrão do arquivo de configuração vimrc para /etc:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Prepare o Vim para compilação:

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

Compile o pacote:

```
make
```

Para preparar os testes, garanta que o(a) usante tester consiga escrever na árvore do fonte e excluir um arquivo contendo testes que exigem **curl** ou **wget**:

```
chown -R tester .
sed '/test_plugin_glvs/d' -i src/testdir/Make_all.mak
```

Agora execute os testes como usante tester:

```
su tester -c "TERM=xterm-256color LANG=en_US.UTF-8 make -j1 test" \
&> vim-test.log
```

A suíte de teste emite um monte de dados binários para a tela. Isso pode causar problemas com as configurações do terminal atual (especialmente enquanto nós estivermos substituindo a variável **TERM** para satisfazer algumas suposições da suíte de teste). O problema pode ser evitado redirecionando-se a saída gerada para um arquivo de registro conforme mostrado acima. Um teste exitoso mostrará **FAILED: 0** no arquivo de registro na conclusão.

Dois testes, **Test_client_server_stopinsert()** e **Test_popup_setbuf()**, são conhecidos por falharem em alguns sistemas.

Instale o pacote:

```
make install
```

Muitos(as) usantes reflexivamente digitam **vi** em vez de **vim**. Para permitir a execução do **vim** quando usantes habitualmente digitarem **vi**, crie um linque simbólico para ambos, o binário e a página de manual, nos idiomas fornecidos:

```
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
```

Por padrão, a documentação do Vim é instalada em **/usr/share/vim**. O seguinte link simbólico permite que a documentação seja acessada via **/usr/share/doc/vim-9.2.1025**, tornando-a consistente com o local da documentação para outros pacotes:

```
ln -sv ../vim/vim92/doc /usr/share/doc/vim-9.2.1025
```

Se um X Window System vai ser instalado no sistema LFS, [então] possivelmente seja necessário recompilar o Vim depois que instalar o X. O Vim vem com uma versão GUI do editor que exige que o X e algumas bibliotecas adicionais seja instalado. Para mais informação a respeito desse processo, consulte a documentação do Vim e a página de instalação do Vim no livro BLFS em <https://www.linuxfromscratch.org/blfs/view/13.1-systemd/postlfs/vim.html>.

8.74.2. Configurando o Vim

Por padrão, **vim** executa em modo incompatível com **vi**. Isso possivelmente seja novo para usantes que tenham usado outros editores no passado. A configuração “*nocompatible*” está incluída abaixo para realçar o fato de que um novo comportamento está sendo usado. Ela também lembra àqueles(as) que mudariam para o modo “*compatible*” que essa deveria ser a primeira configuração no arquivo de configuração. Isso é necessário, pois ela muda outras configurações e substituições precisam vir depois dessa configuração. Crie um arquivo de configuração padrão do **vim** executando o seguinte:

```
cat > /etc/vimrc << "EOF"
" Início do /etc/vimrc

" Certifique-se de que os padrões sejam configurados antes de personalizar as configurações, não depois
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

" Fim do /etc/vimrc
EOF
```

A configuração *set nocompatible* faz com que **vim** se comporte de uma maneira mais útil (o padrão) que a maneira compatível com **vi**. Remova o “no” para manter o comportamento antigo do **vi**. A configuração *set backspace=2* permite retroceder sobre quebras de linha, auto recuos e o início de uma inserção. O parâmetro *syntax on* habilita o destaque de sintaxe do Vim. A configuração *set mouse=* habilita adequada colagem de texto com o mouse quando trabalhar em chroot ou por meio de uma conexão remota. Finalmente, a declaração *if* com a configuração *set background=dark* corrige a suposição do **vim** a respeito da cor de segundo plano de alguns emuladores de terminal. Isso dá ao destaque um esquema de cores melhor para uso no segundo plano preto desses aplicativos.

Documentação para outras opções disponíveis pode ser obtida executando o seguinte comando:

```
vim -c ':options'
```



Nota

Por padrão, o Vim instala somente arquivos de verificador ortográfico para o idioma inglês. Para instalar arquivos de verificador ortográfico para seu idioma preferido, copie os arquivos `.sp1` e, opcionalmente, os `.sug` para seu idioma e codificação de carácter a partir de `runtime/spell` para `/usr/share/vim/vim92/spell/`.

Para usar esses arquivos de verificador ortográfico, alguma configuração em `/etc/vimrc` é necessária, por exemplo:

```
set spelllang=en,ru
set spell
```

Para mais informação, veja-se `runtime/spell/README.txt`.

8.74.3. Conteúdo do Vim

Programas instalados:	ex (link para vim), rview (link para vim), rvim (link para vim), vi (link para vim), view (link para vim), vim , vimdiff (link para vim), vimtutor e xxd
Diretório instalado:	/usr/share/vim

Descrições Curtas

ex	Inicia vim em modo ex
rview	É uma versão restrita do view ; nenhum comando de shell pode ser iniciado e view não pode ser suspenso
rvim	É uma versão restrita do vim ; nenhum comando de shell pode ser iniciado e vim não pode ser suspenso
vi	Link para vim
view	Inicia vim em modo somente leitura
vim	É o editor
vimdiff	Edita duas ou três versões de um arquivo com vim e exibe diferenças
vimtutor	Ensina as teclas e comandos básicas do vim
xxd	Cria um despejo hexadecimal do arquivo dado; ele também pode realizar a operação inversa, de forma que ele pode ser usado para remendamento de binário

8.75. MarkupSafe-3.0.3

MarkupSafe é um módulo do Python que implementa uma sequência de caracteres segura de marcação XML/HTML/XHTML.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 696 KB

8.75.1. Instalação do MarkupSafe

Compile MarkupSafe com o seguinte comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
pip3 install --no-index --find-links dist Markupsafe
```

8.75.2. Conteúdo do MarkupSafe

Diretório instalado: /usr/lib/python3.14/site-packages/MarkupSafe-3.0.3.dist-info

8.76. Jinja2-3.1.6

Jinja2 é um módulo do Python que implementa uma linguagem simples de modelo pitônico.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 2,7 MB

8.76.1. Instalação do Jinja2

Construa o pacote:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instale o pacote:

```
pip3 install --no-index --find-links dist Jinja2
```

8.76.2. Conteúdo do Jinja2

Diretório instalado: /usr/lib/python3.14/site-packages/Jinja2-3.1.6.dist-info

8.77. Systemd-261.2

O pacote systemd contém aplicativos para controlar a inicialização, execução e desligamento do sistema.

Tempo aproximado de 1,2 UPC

construção:

Espaço em disco exigido: 396 MB

8.77.1. Instalação do systemd

Remova dois grupos desnecessários, render e sgx, das regras padrão do "udev":

```
sed -e 's/GROUP="render"/GROUP="video"/' \
    -e 's/GROUP="sgx", //' \
    -i rules.d/50-udev-default.rules.in
```

Prepare systemd para compilação:

```
mkdir -p build
cd      build

meson setup .. \
  --prefix=/usr \
  --buildtype=release \
  -D default-dnssec=no \
  -D firstboot=false \
  -D install-tests=false \
  -D ldconfig=false \
  -D sysusers=false \
  -D rpmmacrodir=no \
  -D homed=disabled \
  -D man=disabled \
  -D mode=release \
  -D pamconfdir=no \
  -D dev-kvm-mode=0660 \
  -D nobody-group=nogroup \
  -D sysupdate=disabled \
  -D ukify=disabled \
  -D docdir=/usr/share/doc/systemd-261.2
```

O significado das opções do meson:

--buildtype=release

Essa chave substitui o tipo de construção padrão (“debug”), que produz binários não otimizados.

-D default-dnssec=no

Essa chave desliga o suporte experimental a DNSSEC.

-D firstboot=false

Essa chave impede a instalação de serviços do systemd responsáveis por configurar o sistema pela primeira vez. Eles não são úteis no LFS, pois tudo é feito manualmente.

-D install-tests=false

Essa chave impede a instalação dos testes compilados.

-D ldconfig=false

Essa chave impede a instalação de uma unidade do systemd que executa **ldconfig** ao inicializar; isso não é útil para distribuições de fonte, tais como o LFS, e torna o tempo de inicialização mais longo. Remova essa opção para habilitar a execução de **ldconfig** ao inicializar.

-D sysusers=false

Essa chave impede a instalação de serviços do systemd responsáveis por configurar os arquivos `/etc/group` e `/etc/passwd`. Ambos os arquivos foram criados no capítulo anterior. Esse processo de segundo plano não é útil em um sistema LFS dado que contas de usante são criadas manualmente.

`-D rpmmacrosdir=no`

Essa chave desabilita a instalação das Macros RPM para uso com o systemd, pois o LFS não suporta RPM.

`-D homed=disabled`

Remova um processo de segundo plano que tem dependências que não se encaixam no escopo do LFS.

`-D man=disabled`

Impeça a geração de páginas de manual para evitar dependências extras. Nós instalaremos páginas de manual pré-geradas para o systemd a partir de um tarball.

`-D mode=release`

Desabilite alguns recursos considerados experimentais pelo(a) desenvolvedor(a).

`-D pamconfdir=no`

Evita a instalação de um arquivo de configuração do PAM não funcional no LFS.

`-D dev-kvm-mode=0660`

A regra padrão do udev permitiria que todos(as) usantes acessassem `/dev/kvm`. Os(As) editores(as) a consideram perigosa. Essa opção a substitui.

`-D nobody-group=nogroup`

Informe ao pacote que o nome do grupo com GID 65534 é nogroup.

`-D sysupdate=disabled`

Não instale a ferramenta "**systemd-sysupdate**". Ela foi projetada para atualizar automaticamente distribuições binárias, de forma que é inútil para um sistema básico do Linux construído a partir do fonte. E informará erros na inicialização se estiver habilitada, mas não configurada corretamente.

`-D ukify=disabled`

Não instale o conjunto de comandos sequenciais "**systemd-ukify**". Em tempo de execução, esse conjunto de comandos sequenciais exige o módulo do "Python" "pefile" que nem o LFS nem o BLFS fornecem.

Compile o pacote:

```
ninja
```

Um teste cria um ponto de montagem no `/tmp` que nós não conseguimos eliminar tão facilmente depois de executar a suíte de teste, e alguns testes precisam de um arquivo `/etc/os-release` básico. Para testar os resultados, crie esse arquivo e execute a suíte de teste em um espaço separado de nomes de montagem (para que o ponto de montagem fique visível somente para a suíte de teste e seja eliminado automaticamente depois do término da suíte de teste):

```
echo 'NAME="Linux From Scratch"' > /etc/os-release
unshare -m ninja test
```

Três testes são conhecidos por falharem no ambiente chroot do LFS, mas passam em uma instalação completa: `core` - `systemd:test-namespace`, `test` - `systemd:test-chase` e `tmpfiles` - `systemd:test-systemd-tmpfiles`. Alguns testes adicionais possivelmente falhem porque eles dependem de várias opções de configuração do núcleo. O teste nominado `test` - `systemd:test-copy` possivelmente exceda o tempo limite devido à congestão de E/S com um grande número de tarefas paralelas, mas passará se executado isoladamente com **meson test test-copy**.

Instale o pacote:

```
ninja install
```

Instale as páginas de manual:

```
tar -xf ../../systemd-man-pages-261.2.tar.xz \
--no-same-owner --strip-components=1 \
-C /usr/share/man
```

Crie o arquivo `/etc/machine-id` necessitado pelo **systemd-journald**:

```
systemd-machine-id-setup
```

Configure a estrutura básica alvo:

```
systemctl preset-all
```

8.77.2. Conteúdo do systemd

Programas instalados:	bootctl, busctl, coredumpctl, halt (linque simbólico para systemctl), hostnamectl, init, journalctl, kernel-install, localectl, loginctl, machinectl, mount.ddi (linque simbólico para systemd-dissect), networkctl, oomctl, portablectl, poweroff (linque simbólico para systemctl), reboot (linque simbólico para systemctl), resolvconf (linque simbólico para resolvectl), resolvectl, run0 (linque simbólico para systemd-run), runlevel (linque simbólico para systemctl), shutdown (linque simbólico para systemctl), systemctl, systemd-ac-power, systemd-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-confext (linque simbólico para systemd-sysex), systemd-creds, systemd-delta, systemd-detect-virt, systemd-dissect, systemd-escape, systemd-hwdb, systemd-id128, systemd-inhibit, systemd-machine-id-setup, systemd-mount, systemd-notify, systemd-nspawn, systemd-path, systemd-pty-forward, systemd-repart, systemd-resolve (linque simbólico para resolvectl), systemd-run, systemd-socket-activate, systemd-stdio-bridge, systemd-sysex, systemd-tmpfiles, systemd-tty-ask-password-agent, systemd-vpick, systemd-umount (linque simbólico para systemd-mount), timedatectl, udevadm, userdbctl e varlinkctl
Bibliotecas instaladas:	libnss_myhostname.so.2, libnss_mymachines.so.2, libnss_resolve.so.2, libnss_systemd.so.2, libsystemd.so, libsystemd-shared-261.2.so (em <code>/usr/lib/systemd</code>) e libudev.so
Diretórios instalados:	<code>/etc/binfmt.d</code> , <code>/etc/init.d</code> , <code>/etc/kernel</code> , <code>/etc/modules-load.d</code> , <code>/etc/sysctl.d</code> , <code>/etc/systemd</code> , <code>/etc/tmpfiles.d</code> , <code>/etc/udev</code> , <code>/etc/xdg/systemd</code> , <code>/usr/include/systemd</code> , <code>/usr/lib/binfmt.d</code> , <code>/usr/lib/credstore</code> , <code>/usr/lib/environment.d</code> , <code>/usr/lib/kernel</code> , <code>/usr/lib/modprobe.d</code> , <code>/usr/lib/modules-load.d</code> , <code>/usr/lib/systemd</code> , <code>/usr/lib/udev</code> , <code>/usr/lib/sysctl.d</code> , <code>/usr/lib/systemd</code> , <code>/usr/lib/tmpfiles.d</code> , <code>/usr/share/doc/systemd-261.2</code> , <code>/usr/share/factory</code> , <code>/usr/share/systemd</code> , <code>/var/lib/systemd</code> e <code>/var/log/journal</code>

Descrições Curtas

bootctl	É usado para controlar configurações de inicialização do firmware EFI em um sistema
busctl	É usado para introspecção e monitoramento do barramento D-Bus
coredumpctl	É usado para recuperar despejos da memória principal a partir do diário do systemd
halt	Normalmente invoca shutdown com a opção <code>-h</code> , exceto quando já em nível de execução 0, então ele informa ao núcleo para parar o sistema; ele anota no arquivo <code>/var/log/wtmp</code> que o sistema está sendo desligado
hostnamectl	É usado para consultar e mudar o nome de dispositivo do sistema e configurações relacionadas
init	É o primeiro processo a ser iniciado depois que o núcleo tenha inicializado o hardware; init assume o processo de inicialização e

	inicia os processos especificados pelos arquivos de configuração dele; nesse caso, ele inicia o systemd
journalctl	É usado para consultar o conteúdo do diário do systemd
kernel-install	É usado para adicionar e remover imagens de núcleo e de initramfs para e de /boot; no LFS, isso é feito manualmente
localectl	É usado para consultar e mudar as configurações de locale e esquema de teclado do sistema
loginctl	É usado para introspectar e controlar o estado do Login Manager do systemd
machinectl	É usado para introspectar e controlar o estado do Virtual Machine e do Container Registration Manager do systemd
networkctl	É usado para introspectar e configurar o estado dos links da rede de comunicação configurados pelo systemd-networkd
oomctl	Controla o processo de segundo plano Out Of Memory do systemd
portablectl	É usado para anexar ou desanexar serviços portáteis a partir do sistema local
poweroff	Instrui o núcleo a parar o sistema e desligar o computador (veja-se halt)
reboot	Instrui o núcleo a reinicializar o sistema (veja-se halt)
resolvconf	Registra a configuração do servidor de DNS e domínio com systemd-resolved
resolvectl	Envia comandos de controle para o gerenciador de resolução de nome da rede de comunicação ou resolve nomes de domínio, endereços IPv4 e IPv6, registros DNS e serviços
run0	Temporariamente eleva ou adquire privilégios diferentes; similar ao sudo
runlevel	Emite o nível de execução anterior e o atual, conforme anotado no registro mais recente do nível de execução em /run/utmp
shutdown	Desativa o sistema de uma maneira segura e protegida, sinalizando para todos os processos e notificando todos(as) os(as) usantes logados(as)
systemctl	É usado para introspectar e controlar o estado do gerenciador de sistema e serviço do systemd
systemd-ac-power	Informa se o sistema está conectado a uma fonte externa de eletricidade.
systemd-analyze	É usado para analisar o desempenho da inicialização do sistema, bem como para identificar unidades problemáticas do systemd
systemd-ask-password	É usado para consultar uma palavra secreta ou frase secreta do sistema a partir do(a) usante, usando uma mensagem especificada na linha de comando do Linux
systemd-cat	É usado para conectar as saídas geradas STDOUT e STDERR de um processo com o diário do systemd
systemd-cgls	Recursivamente mostra o conteúdo da hierarquia do grupo de controle selecionado do Linux em uma árvore

systemd-cgtop	Mostra os grupos de controle superiores da hierarquia do grupo de controle local do Linux, ordenados pela carga de CPU, memória e entradas/saídas de disco deles
systemd-creds	Exibe e processa credenciais
systemd-delta	É usado para identificar e comparar arquivos de configuração em / etc que substituem os padrões em /usr
systemd-detect-virt	Detecta se o sistema está sendo executado em um ambiente virtual e ajusta o Udev de acordo
systemd-dissect	É usado para inspecionar imagens de disco de SO
systemd-escape	É usado para escapar sequências de caracteres para inclusão em nomes de unidade do systemd
systemd-hwdb	É usado para gerenciar a base de dados de hardware (hwdb)
systemd-id128	Gera e imprime sequências de caracteres id128 (UUID)
systemd-inhibit	É usado para executar um aplicativo com um bloqueio de inibição de desligamento, suspensão ou inatividade, impedindo uma ação tal como um desligamento do sistema até que o processo seja completado
systemd-machine-id-setup	É usado por ferramentas de instalador do sistema para inicializar o ID de máquina armazenado em /etc/machine-id ao tempo de instalação com um ID gerado aleatoriamente
systemd-mount	É usado para montar temporariamente ou automontar discos
systemd-notify	É usado por scripts do processo de segundo para notificar o sistema init acerca de mudanças de situação
systemd-nspawn	É usado para executar um comando ou um OS inteiro, em um contêiner de espaço de nome peso leve
systemd-path	É usado para consultar caminhos de sistema e usante
systemd-pty-forward	É usado para executar um comando com uma cor personalizada de plano de fundo do terminal, ou título personalizado
systemd-repart	É usado para aumentar e adicionar partições à uma tabela de partição quando o systemd for usado com uma imagem de SO (por exemplo, um contêiner)
systemd-resolve	É usado para resolver nomes de domínio, endereços IPV4 e IPV6, registros de recurso DNS e serviços
systemd-run	É usado para criar e iniciar uma unidade transitória .service ou uma .scope e executar o comando especificado nela; isso é útil para validar unidades do systemd
systemd-socket-activate	É usado para ouvir em dispositivos de soquete e iniciar um processo depois de uma conexão bem sucedida com o soquete
systemd-sysex	Ativa imagens de extensão de sistema
systemd-tmpfiles	Cria, deleta e limpa arquivos e diretórios voláteis e temporários, baseado no formato de arquivo de configuração e local especificado em diretórios tmpfiles.d
systemd-umount	Desmonta pontos de montagem

systemd-tty-ask-password-agent	É usado para listar e (ou) processar solicitações pendentes de senha do systemd
systemd-vpick	É usado para resolver caminhos para um diretório versionado ".v/"
timedatectl	É usado para consultar e mudar o relógio do sistema e as configurações dele
udevadm	É uma ferramenta genérica de administração do Udev que controla o processo de segundo plano udevd, fornece informação a partir da base de dados de hardware do Udev, monitora uevents, aguarda que uevents finalizem, testa a configuração do Udev e deflagra uevents para um dispositivo dado
userdbctl	É usado para inspecionar usantes, grupos e associações de grupos
varlinkctl	É usado para interagir com, e invocar, serviços Varlink
libsystemd	É a biblioteca principal de utilitário do systemd
libudev	É uma biblioteca para acessar informação de dispositivo do Udev

8.78. D-Bus-1.16.2

D-Bus é um sistema de barramento de mensagem, uma maneira simples para aplicativos conversarem entre si. D-Bus fornece ambos, um processo de segundo plano de sistema (para eventos tais como "novo dispositivo de hardware adicionado" ou "fila de impressora mudou") e um processo de segundo plano de sessão de login de usante (para necessidades gerais de IPC entre aplicativos de usante). Também, o barramento de mensagem é construído no topo de uma estrutura geral de envio de mensagem um-a-um, a qual pode ser usada por quaisquer dois aplicativos para se comunicarem diretamente (sem transitar pelo processo de segundo plano do barramento de mensagem).

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 17 MB

8.78.1. Instalação do D-Bus

Prepare D-Bus para compilação:

```
mkdir build
cd build

meson setup --prefix=/usr --buildtype=release --wrap-mode=nofallback ..
```

O significado das opções do meson:

`--wrap-mode=nofallback`

Essa chave impede que o meson tente baixar uma cópia do pacote Glib para os testes.

Compile o pacote:

```
ninja
```

Para testar os resultados, emita:

```
ninja test
```

Muitos testes são desabilitados, pois eles exigem pacotes adicionais que não estão inclusos no LFS. Instruções para executar a suíte completa de teste podem ser encontradas no *livro BLFS*.

Instale o pacote:

```
ninja install
```

Crie um link simbólico, de forma que D-Bus e systemd possam usar o mesmo arquivo `machine-id`:

```
ln -sfv /etc/machine-id /var/lib/dbus
```

8.78.2. Conteúdo do D-Bus

Programas instalados:	dbus-cleanup-sockets, dbus-daemon, dbus-launch, dbus-monitor, dbus-run-session, dbus-send, dbus-test-tool, dbus-update-activation-environment e dbus-uuidgen
Bibliotecas instaladas:	libdbus-1.so
Diretórios instalados:	/etc/dbus-1, /usr/include/dbus-1.0, /usr/lib/dbus-1.0, /usr/share/dbus-1, /usr/share/doc/dbus-1.16.2 e /var/lib/dbus

Descrições Curtas

dbus-cleanup-sockets	é usado para remover soquetes restantes em um diretório
dbus-daemon	É o processo de segundo plano do barramento de mensagem do D-Bus

dbus-launch	inicia dbus-daemon a partir de um conjunto de comandos sequenciais de shell
dbus-monitor	monitora mensagens transitando ao longo de um barramento de mensagem do D-Bus
dbus-run-session	inicia uma instância de barramento de sessão do dbus-daemon a partir de um conjunto de comandos sequenciais de shell e inicia um programa especificado naquela sessão
dbus-send	envia uma mensagem para um barramento de mensagem do D-Bus
dbus-test-tool	é uma ferramenta para auxiliar pacotes a testarem o D-Bus
dbus-update-activation-environment	atualiza variáveis de ambiente que serão configuradas para serviços de sessão do D-Bus
dbus-uuidgen	Gera um ID único universalmente
libdbus-1	Contém funções de API usadas para comunicar com o barramento de mensagem do D-Bus

8.79. Man-DB-2.13.1

O pacote Man-DB contém aplicativos para encontrar e visualizar páginas de manual.

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 44 MB

8.79.1. Instalação do Man-DB

Prepare Man-DB para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/man-db-2.13.1 \
            --sysconfdir=/etc \
            --disable-setuid \
            --enable-cache-owner=bin \
            --with-browser=/usr/bin/lynx \
            --with-vgrind=/usr/bin/vgrind \
            --with-grap=/usr/bin/grap
```

O significado das opções do configure:

`--disable-setuid`

Isso desabilita tornar o programa **man** setuid para o(a) usante man.

`--enable-cache-owner=bin`

Isso muda a titularidade de propriedade dos arquivos de cache abrangentes ao sistema para o(a) usante bin.

`--with-...`

Esses três parâmetros são usados para configurar alguns aplicativos padrão. **lynx** é um navegador da web baseado em texto (veja-se o BLFS para instruções de instalação); **vgrind** converte fontes de aplicativo para entrada gerada do Groff; e **grap** é útil para tipografar gráficos em documentos do Groff. Os aplicativos **vgrind** e **grap** normalmente não são necessários para visualizar páginas de manual. Eles não são parte do LFS ou do BLFS, mas você deveria ser capaz de instalá-los você mesmo(a) depois de terminar o LFS, se desejar fazer isso.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.79.2. Páginas de Manual não inglesas no LFS

A seguinte tabela mostra o conjunto de caracteres que o Man-DB supõe que as páginas de manual instaladas sob `/usr/share/man/<11>` estarão codificadas. Em adição a isso, o Man-DB determina corretamente se páginas de manual instaladas naquele diretório estão codificadas em UTF-8.

Tabela 8.1. Codificação de caracteres esperada das páginas de manual legadas de 8 bits

Idioma (código)	Codificação	Idioma (código)	Codificação
Dinamarquês (da)	ISO-8859-1	Croata (hr)	ISO-8859-2
Alemão (de)	ISO-8859-1	Húngaro (hu)	ISO-8859-2
Inglês (en)	ISO-8859-1	Japonês (ja)	EUC-JP

Idioma (código)	Codificação	Idioma (código)	Codificação
Espanhol (es)	ISO-8859-1	Coreano (ko)	EUC-KR
Estoniano (et)	ISO-8859-1	Lituano (lt)	ISO-8859-13
Finlandês (fi)	ISO-8859-1	Letão (lv)	ISO-8859-13
Francês (fr)	ISO-8859-1	Macedônio (mk)	ISO-8859-5
Irlandês (ga)	ISO-8859-1	Polonês (pl)	ISO-8859-2
Galego (gl)	ISO-8859-1	Romeno (ro)	ISO-8859-2
Indonésio (id)	ISO-8859-1	Grego (el)	ISO-8859-7
Islandês (is)	ISO-8859-1	Eslovaco (sk)	ISO-8859-2
Italiano (it)	ISO-8859-1	Esloveno (sl)	ISO-8859-2
Bokmal norueguês (nb)	ISO-8859-1	Latim sérvio (sr@latin)	ISO-8859-2
Holandês (nl)	ISO-8859-1	Sérvio (sr)	ISO-8859-5
Nynorsk norueguês (nn)	ISO-8859-1	Turco (tr)	ISO-8859-9
Norueguês (no)	ISO-8859-1	Ucraniano (uk)	KOI8-U
Português (pt)	ISO-8859-1	Vietnamita (vi)	TCVN5712-1
Sueco (sv)	ISO-8859-1	Chinês simplificado (zh_CN)	GBK
Bielorrusso (be)	CP1251	Chinês simplificado, Singapura (zh_SG)	GBK
Búlgaro (bg)	CP1251	Chinês tradicional, Hong Kong (zh_HK)	BIG5HKSCS
Tcheco (cs)	ISO-8859-2	Chinês tradicional (zh_TW)	BIG5



Nota

Páginas de manual em idiomas que não estão na lista não são suportadas.

8.79.3. Conteúdo do Man-DB

Programas instalados: accessdb, apropos (link para whatis), catman, lexgrog, man, man-recode, mandb, manpath e whatis

Bibliotecas instaladas: libman.so e libmandb.so (ambas em /usr/lib/man-db)

Diretórios instalados: /usr/lib/man-db, /usr/libexec/man-db e /usr/share/doc/man-db-2.13.1

Descrições Curtas

accessdb Despeja o conteúdo da base de dados **whatis** em formato legível por humanos

apropos Pesquisa na base de dados **whatis** e exibe as descrições curtas dos comandos de sistema que contém uma sequência de caracteres dada

catman Cria ou atualiza páginas de manual pré-formatadas

lexgrog Exibe informação de sumário em uma linha a respeito de uma página de manual dada

man Formata e exibe a página de manual solicitada

man-recode Converte páginas de manual para outra codificação

mandb	Cria ou atualiza a base de dados whatis
manpath	Exibe o conteúdo de \$MANPATH ou (se \$MANPATH não estiver configurada) um caminho de busca adequado baseado nas configurações em man.conf e no ambiente do(a) usante
whatis	Pesquisa na base de dados whatis e exibe as descrições curtas de comandos do sistema que contenham a palavra chave dada como uma palavra separada
libman	Contém suporte em tempo de execução para o man
libmandb	Contém suporte em tempo de execução para o man

8.80. Procps-ng-4.0.7

O pacote Procps-ng contém aplicativos para monitorar processos.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 28 MB

8.80.1. Instalação do Procps-ng

Prepare procps-ng para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/procps-ng-4.0.7 \
            --disable-static \
            --disable-kill \
            --enable-watch8bit \
            --with-systemd
```

O significado da opção de configure:

--disable-kill

Essa chave desabilita a construção do comando **kill**; ele será instalado a partir do pacote Util-linux.

--enable-watch8bit

Essa chave habilita o suporte ncursesw para o comando **watch**, de forma que ele consiga manusear caracteres de 8 bits.

Compile o pacote:

```
make
```

Para executar a suíte de teste, execute:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Um teste chamado `ps` com sinalizador de saída `bsdtime`, `cputime`, `etime`, `etimes` é conhecido por falhar se o núcleo do anfitrião não for construído com `CONFIG_BSD_PROCESS_ACCT` habilitado.

Instale o pacote:

```
make install
```

8.80.2. Conteúdo do Procps-ng

Programas instalados:	free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w e watch
Biblioteca instalada:	libproc-2.so
Diretórios instalados:	/usr/include/procps e /usr/share/doc/procps-ng-4.0.7

Descrições Curtas

free	Informa a quantidade de memória livre e usada (ambas memória física e de troca) no sistema
pgrep	Procura por processos baseado nos nomes deles e outros atributos
pidof	Informa os PIDs dos aplicativos dados
pkill	Sinaliza processos baseado nos nomes deles e outros atributos
pmap	Informa o mapeamento de memória do processo dado
ps	Lista os processos em execução atualmente

pwdx	Informa o diretório atual de trabalho de um processo
slabtop	Exibe informação detalhada do cache slab do núcleo em tempo real
sysctl	Modifica parâmetros do núcleo em tempo de execução
tload	Imprime um gráfico da média atual da carga do sistema
top	Exibe uma lista dos processos mais intensivos da CPU; ele fornece uma visão contínua da atividade do processador em tempo real
uptime	Informa há quanto tempo o sistema tem estado executando, quantos(as) usantes estão logados(as) e as médias de carga do sistema
vmstat	Informa estatísticas de memória virtual, dando informação a respeito de processos, memória, paginação, Entrada/Saída (E/S) de bloco, armadilhas e atividade da CPU
w	Mostra quais usantes estão atualmente logados(as), onde e desde quando
watch	Executa um comando dado repetidamente, exibindo a primeira tela cheia da saída gerada dele; isso permite que um(a) usante observe a mudança de saída gerada ao longo do tempo
libproc-2	Contém as funções usadas pela maioria dos aplicativos nesse pacote

8.81. Util-linux-2.42.2

O pacote Util-linux contém aplicativos utilitários diversos. Entre eles estão utilitários para lidar com sistemas de arquivos, consoles, partições e mensagens.

Tempo aproximado de construção: 0,5 UPC

Espaço em disco exigido: 362 MB

8.81.1. Instalação do Util-linux

Prepare o Util-linux para compilação:

```
./configure --bindir=/usr/bin      \
            --libdir=/usr/lib      \
            --runstatedir=/run     \
            --sbindir=/usr/sbin   \
            --disable-chfn-chsh    \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-liblastlog2 \
            --disable-static      \
            --without-python      \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.42.2
```

As opções `--disable` e `--without` impedem avisos acerca de construir componentes que ou exigem pacotes ausentes no LFS ou são inconsistentes com aplicativos instalados por outros pacotes.

Compile o pacote:

```
make
```

Se desejado, crie um arquivo fictício `/etc/fstab` para satisfazer dois testes e execute a suíte de teste como um(a) usuário não-root:



Atenção

Executar a suíte de teste como o(a) usuário root pode ser danoso para teu sistema. Para executá-lo, a opção `CONFIG_SCSI_DEBUG` para o núcleo precisa estar disponível no sistema em execução atualmente e precisa ser construída como um módulo. Construí-la dentro do núcleo impedirá a inicialização. Para cobertura completa, outros pacotes do BLFS precisam ser instalados. Se desejado, esse teste pode ser executado reiniciando-se no sistema completo LFS e executando:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
touch /etc/fstab
chown -R tester .
su tester -c "make -k check"
```

O `lsfd`: teste `inotify` falhará se a opção do núcleo `CONFIG_NETLINK_DIAG` não estiver habilitada.

Instale o pacote:

```
make install
```

8.81.2. Conteúdo do Util-linux

Programas instalados:	addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, findcore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hardlink, hexdump, hwclock, i386 (link para setarch), ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (link para last), ldattach, linux32 (link para setarch), linux64 (link para setarch), logger, look, losetup, lsblk, lscpu, lsipc, lsirq, lsfd, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfdisk, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, taskset, uclampset, ul, umount, uname26 (link para setarch), unshare, utmpdump, uuuid, uuidgen, uuidparse, wall, wdctl, whereis, wpefs, x86_64 (link para setarch) e zramctl
Bibliotecas instaladas:	libblkid.so, libfdisk.so, libmount.so, libsmartcols.so e libuuid.so
Diretórios instalados:	/usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.42.2 e /var/lib/hwclock

Descrições Curtas

addpart	Informa ao núcleo Linux das novas partições
agetty	Abre uma porta tty, solicita um nome de login e então invoca o aplicativo login
blkdiscard	Descarta setores em um dispositivo
blkid	Um utilitário de linha de comando para localizar e imprimir atributos de dispositivo de bloco
blkzone	É usado para gerenciar dispositivos zoneados de armazenamento de bloco
blockdev	Permite a usantes chamarem ioctls de dispositivo de bloco a partir da linha de comando
cal	Exibe um calendário simples
cfdisk	Manipula a tabela de partição do dispositivo dado
chcpu	Modifica o estado de CPUs
chmem	Configura memória
choom	Exibe e ajusta placares OOM-Killer, usados para determinar quais processos matar primeiro quando o Linux estiver Fora da Memória
chrt	Manipula atributos de tempo real de um processo
col	Filtra feeds de linha reversa
colcrt	Filtra saída gerada do nroff para terminais que carecem de alguns recursos, tais como overstriking e half-lines
colrm	Filtra as colunas dadas
column	Formata um arquivo dado em colunas múltiplas
ctrlaltdel	Configura a função da combinação de teclas Ctrl+Alt+Del para uma reconfiguração rígida ou uma flexível
delpart	Pede ao núcleo Linux para remover uma partição
dmesg	Despeja as mensagens de inicialização do núcleo
eject	Ejeta mídia removível
fallocate	Pré-aloca espaço para um arquivo

fdisk	Manipula a tabela de partição do dispositivo dado
fincore	Conta páginas de conteúdo de arquivo em núcleo
findfs	Encontra um sistema de arquivos, ou pelo rótulo ou pelo Universally Unique Identifier (UUID)
findmnt	É uma interface de linha de comando para a biblioteca libmount para funcionar com mountinfo, fstab e arquivos mtab
flock	Adquire uma trava de arquivo e então executa um comando com a trava mantida
fsck	É usado para verificar, e opcionalmente reparar, sistemas de arquivos
fsck.cramfs	Realiza uma verificação de consistência no sistema de arquivos Cramfs no dispositivo dado
fsck.minix	Realiza uma verificação de consistência no sistema de arquivos Minix no dispositivo dado
fsfreeze	É um encapsulador muito simples em torno de operações FIFREEZE/FITHAW do controlador ioctl de núcleo
fstrim	Descarta blocos não usados em um sistema de arquivos montado
getopt	Analisa opções na linha de comando dada
hardlink	Consolida arquivos duplicados criando links rígidos
hexdump	Despeja o arquivo dado em hexadecimal, decimal, octal ou ascii
hwclock	Lê ou configura o relógio de hardware do sistema, também chamado de Real-Time Clock (RTC) ou de relógio do Basic Input-Output System (BIOS)
i386	Um link simbólico para setarch
ionice	Obtém ou configura a classe de agendamento de E/S e prioridade para um programa
ipcmk	Cria vários recursos IPC
ipcrm	Remove o dado recurso de Inter-Process Communication (IPC)
ipcs	Fornecer informação de situação de IPC
irqtop	Exibe informação de contador de interrupção do núcleo em visão estilo " <i>top(1)</i> "
isozsize	Informa o tamanho de um sistema de arquivos iso9660
kill	Envia sinais para processos
last	Mostra quais usantes logaram-se (e deslogaram-se) mais recentemente, pesquisando de volta ao longo do arquivo <code>/var/log/wtmp</code> ; ele também mostra inicializações do sistema, desligamentos e mudanças de nível de execução
lastb	Exibe as tentativas falhas de login, conforme registrado em <code>/var/log/btmp</code>
ldattach	Anexa uma disciplina de linha à uma linha serial
linux32	Um link simbólico para setarch
linux64	Um link simbólico para setarch
logger	Adiciona a mensagem dada ao registro do sistema
look	Exibe linhas que começam com a sequência de caracteres dada
losetup	Configura e controla dispositivos de loop
lsblk	Lista informação a respeito de todos ou de dispositivos de bloco selecionados em um formato semelhante a árvore
lscpu	Imprime informação de arquitetura da CPU
lsfd	Exibe informação a respeito de arquivos abertos; substitui o ls
lsipc	Imprime informação acerca de facilidades de IPC empregadas atualmente no sistema

lsirq	Exibe informação do contador de interrupção do núcleo
lslocks	Lista travas locais de sistema
lslogins	Lista informação acerca de contas de usantes, de grupos e de sistema
lsmem	Lista os intervalos de memória disponível com a situação online deles
lsns	Lista espaços de nome
mcookie	Gera cookies mágicos (números hexadecimais aleatórios de 128 bits) para o xauth
mesg	Controla se outros(as) usantes podem enviar mensagens para o terminal do(a) usante atual
mkfs	Constrói um sistema de arquivos em um dispositivo (geralmente uma partição de disco rígido)
mkfs.bfs	Cria um sistema de arquivos Santa Cruz Operations (SCO) bfs
mkfs.cramfs	Cria um sistema de arquivos cramfs
mkfs.minix	Cria um sistema de arquivos Minix
mkswap	Inicializa dispositivo ou arquivo dado para ser usado como uma área de troca
more	Um filtro para paginar ao longo de texto uma tela de cada vez
mount	Anexa o sistema de arquivos no dispositivo dado a um diretório especificado na árvore do sistema de arquivos
mountpoint	Verifica se o diretório é um ponto de montagem
namei	Mostra os links simbólicos nos caminhos dados
nsenter	Executa um aplicativo com espaços de nome de outros processos
partx	Informa ao núcleo a respeito da presença e numeração de partições no disco
pivot_root	Torna o sistema de arquivos dado o novo sistema de arquivos raiz do processo atual
prlimit	Obtém e configura um limite de recursos do processo
readprofile	Lê informação de perfil do núcleo
rename	Renomeia os arquivos dados, substituindo uma sequência de caracteres dada por outra
renice	Altera a prioridade de processos em execução
resizepart	Pede ao núcleo Linux para redimensionar uma partição
rev	Inverte as linhas de um arquivo dado
rfskill	Ferramenta para habilitar e desabilitar dispositivos sem fios
rtcwake	Usado para entrar em um estado de suspensão do sistema até o horário de ativação especificado
script	Cria um texto datilografado de uma sessão de terminal
scriptlive	Reexecuta textos datilografados de sessão usando informação de tempo
scriptreplay	Reproduz textos datilografados usando informação de tempo
setarch	Muda a arquitetura informada em um novo ambiente de aplicativo e configura sinalizadores de personalidade
setsid	Executa o aplicativo dado em uma nova sessão
setterm	Configura atributos do terminal
sfdisk	Um manipulador de tabela de partição de disco
sulogin	Permite root se logar; ele normalmente é invocado por init quando o sistema entra em modo de usante único(a)
swapon	Faz mudanças para o UUID e rótulo da área de troca

swapon	Desabilita dispositivos e arquivos para paginação e troca
swapoff	Habilita dispositivos e arquivos para paginação e troca e lista os dispositivos e arquivos atualmente em uso
switch_root	Altera para outro sistema de arquivos como a raiz da árvore de montagem
taskset	Recupera ou configura uma afinidade de CPU do processo
uclampset	Manipula os atributos de fixação de utilização do sistema ou de um processo
ul	Um filtro para traduzir sublinhados em sequências de escape indicando sublinhamento para o terminal em uso
umount	Desconecta um sistema de arquivos da árvore de arquivos do sistema
uname26	Um link simbólico para setarch
unshare	Executa um aplicativo com alguns espaços de nome não compartilhados oriundos do(a) ancestral
utmpdump	Exibe o conteúdo do arquivo de login dado em um formato amigável para o(a) usante
uudd	Um processo de segundo plano usado pela biblioteca UUID para gerar UUIDs baseados em horário em uma forma segura e garantidamente única
uuidgen	Cria novos "UUIDs". Cada novo "UUID" é um número aleatório considerado ser único entre todos os "UUIDs" criados, no sistema local e em outros sistemas, no passado e no futuro, com probabilidade extremamente alta (2^{128} "UUIDs" são possíveis)
uuidparse	Um utilitário para analisar identificadores únicos
wall	Exibe o conteúdo de um arquivo ou, por padrão, a entrada gerada padrão dele, nos terminais de todos(as) os(as) usantes logados(as) atualmente
wdctl	Mostra a situação do vigilante de hardware
whereis	Informa o local do binário, fonte e arquivos de página de manual para o comando dado
wipefs	Limpa uma assinatura de sistema de arquivos a partir de um dispositivo
x86_64	Um link simbólico para setarch
zramctl	Um aplicativo para configurar e controlar dispositivos zram (disco ram comprimido)
libblkid	Contém rotinas para identificação de dispositivo e extração de token
libfdisk	Contém rotinas para manipular tabelas de partição
libmount	Contém rotinas para montagem e desmontagem de dispositivo de bloco
libsmartcols	Contém rotinas para auxiliar a saída gerada de tela em forma tabular
libuuid	Contém rotinas para gerar identificadores únicos para objetos que possivelmente sejam acessíveis além do sistema local

8.82. E2fsprogs-1.47.4

O pacote E2fsprogs contém os utilitários para lidar com o sistema de arquivos ext2. Ele também suporta os sistemas de arquivos de registro em diário ext3 e ext4.

Tempo aproximado de construção: 2,4 UPC em um disco rotatório, 0,4 UPC em um SSD

Espaço em disco exigido: 101 MB

8.82.1. Instalação do E2fsprogs

A documentação do E2fsprogs recomenda que o pacote seja construído em um subdiretório da árvore do fonte:

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

Prepare E2fsprogs para compilação:

```
../configure --prefix=/usr \
              --sysconfdir=/etc \
              --enable-elf-shlibs \
              --disable-libblkid \
              --disable-libuuid \
              --disable-uuid \
              --disable-fsck
```

O significado das opções do configure:

`--enable-elf-shlibs`

Isso cria as bibliotecas compartilhadas as quais alguns aplicativos nesse pacote usam.

`--disable-*`

Isso evita construir e instalar as bibliotecas libuuid e libblkid, o processo de segundo plano uuidd e o encapsulador **fsck**; util-linux instala versões mais recentes.

Compile o pacote:

```
make
```

Para executar os testes, emita:

```
make check
```

Um teste chamado `m_assume_storage_prezeroed` é conhecido por falhar. Outro teste chamado `m_rootdir_acl` é conhecido por falhar se o sistema de arquivos usado para o sistema LFS não for ext4.

Instale o pacote:

```
make install
```

Remova bibliotecas estáticas inúteis:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Esse pacote instala um arquivo gzipado `.info`, mas não atualiza o arquivo abrangente ao sistema `dir`. Descompacte esse arquivo e então atualize o arquivo do sistema `dir` usando os seguintes comandos:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

Se desejado, crie e instale alguma documentação adicional emitindo os seguintes comandos:

```
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
```

8.82.2. Configurando o E2fsprogs

`/etc/mke2fs.conf` contém o valor padrão de várias opções de linha de comando do **mke2fs**. Você possivelmente edite o arquivo para tornar os valores padrão convenientes para as tuas necessidades. Por exemplo, alguns utilitários (não no LFS ou no BLFS) não conseguem reconhecer um sistema de arquivos `ext4` com o recurso `metadata_csum_seed` habilitado. Se você precisar de tal utilitário, você possivelmente remova o recurso da lista padrão de recurso do `ext4` com o comando:

```
sed 's/metadata_csum_seed, //' -i /etc/mke2fs.conf
```

Leia-se a página de manual "*mke2fs.conf(5)*" para detalhes.

8.82.3. Conteúdo do E2fsprogs

Programas instalados:	<code>badblocks</code> , <code>chattr</code> , <code>compile_et</code> , <code>debugfs</code> , <code>dumpe2fs</code> , <code>e2freefrag</code> , <code>e2fsck</code> , <code>e2image</code> , <code>e2label</code> , <code>e2mmpstatus</code> , <code>e2scrub</code> , <code>e2scrub_all</code> , <code>e2undo</code> , <code>e4crypt</code> , <code>e4defrag</code> , <code>filefrag</code> , <code>fsck.ext2</code> , <code>fsck.ext3</code> , <code>fsck.ext4</code> , <code>logsave</code> , <code>lsattr</code> , <code>mk_cmds</code> , <code>mke2fs</code> , <code>mkfs.ext2</code> , <code>mkfs.ext3</code> , <code>mkfs.ext4</code> , <code>mklost+found</code> , <code>resize2fs</code> e <code>tune2fs</code>
Bibliotecas instaladas:	<code>libcom_err.so</code> , <code>libe2p.so</code> , <code>libext2fs.so</code> e <code>libss.so</code>
Diretórios instalados:	<code>/usr/include/e2p</code> , <code>/usr/include/et</code> , <code>/usr/include/ext2fs</code> , <code>/usr/include/ss</code> , <code>/usr/lib/e2fsprogs</code> , <code>/usr/share/et</code> e <code>/usr/share/ss</code>

Descrições Curtas

badblocks	Pesquisa em um dispositivo (geralmente uma partição de disco) por blocos ruins
chattr	Muda os atributos de arquivos em sistemas de arquivos <code>ext{234}</code>
compile_et	Um compilador de tabela de erro; ele converte uma tabela de nomes e mensagens dos códigos de erros em um arquivo fonte C adequado para uso com a biblioteca <code>com_err</code>
debugfs	Um depurador de sistema de arquivo; ele pode ser usado para examinar e mudar o estado de sistemas de arquivos <code>ext{234}</code>
dumpe2fs	Imprime a informação de superblocos e de grupo de blocos para o sistema de arquivos presente em um dispositivo dado
e2freefrag	Informa informação de fragmentação de espaço livre
e2fsck	É usado para verificar e opcionalmente reparar sistema de arquivos <code>ext{234}</code>
e2image	É usado para salvar dados críticos do sistema de arquivos <code>ext{234}</code> para um arquivo
e2label	Exibe ou muda o rótulo do sistema de arquivos no sistema de arquivos <code>ext{234}</code> em um dispositivo dado
e2mmpstatus	Verifica a situação da MMP (Multiple Mount Protection) de um sistema de arquivos <code>ext4</code>
e2scrub	Verifica o conteúdo de um sistema de arquivos <code>ext{234}</code> montado
e2scrub_all	Verifica todos os sistemas de arquivos <code>ext[234]</code> para erros
e2undo	Repete o registro de desfazer para um sistema de arquivos <code>ext[234]</code> encontrado em um dispositivo. [Isso pode ser usado para desfazer uma operação falha por um programa do E2fsprogs].
e4crypt	Utilitário de encriptação do sistema de arquivos <code>ext4</code>
e4defrag	Desfragmentador em linha para sistema de arquivos <code>ext4</code>
filefrag	Informa o quão mau fragmentado um arquivo específico pode estar
fsck.ext2	Por padrão verifica sistemas de arquivo <code>ext2</code> e é um link rígido para e2fsck
fsck.ext3	Por padrão verifica sistemas de arquivo <code>ext3</code> e é um link rígido para e2fsck

fsck.ext4	Por padrão verifica sistemas de arquivo ext4 e é um link rígido para e2fsck
logsave	Salva a saída gerada de um comando em um arquivo de registro
lsattr	Lista os atributos de arquivos em um sistema de arquivos segundo estendido
mk_cmds	Converte uma tabela de nomes de comando e mensagens de ajuda em um arquivo fonte C adequado para uso com a biblioteca de subsistema libss
mke2fs	Cria um sistema de arquivos ext{234} no dispositivo dado
mkfs.ext2	Por padrão cria sistemas de arquivos ext2 e é um link rígido para mke2fs
mkfs.ext3	Por padrão cria sistemas de arquivos ext3 e é um link rígido para mke2fs
mkfs.ext4	Por padrão cria sistemas de arquivos ext4 e é um link rígido para mke2fs
mklost+found	Cria um diretório lost+found em um sistema de arquivos ext{234}; ele pré-aloca blocos de disco para esse diretório para facilitar a tarefa do e2fsck
resize2fs	Pode ser usado para alargar ou estreitar sistema de arquivos ext{234}
tune2fs	Ajusta parâmetros ajustáveis do sistema de arquivos em sistema de arquivos ext{234}
libcom_err	A rotina comum de exibição de erro
libe2p	Usado por dumpe2fs , chattr e lsattr
libext2fs	Contém rotinas para habilitar programas de nível de usante para manipular sistemas de arquivos ext{234}
libss	Usado por debugfs

8.83. Acerca dos Símbolos de Depuração

A maioria dos programas e bibliotecas é, por padrão, compilado com símbolos de depuração inclusos (com a opção `-g` do `gcc`). Isso significa que ao depurar um programa ou biblioteca que foi compilado com informação de depuração, o depurador consegue fornecer não apenas endereços de memória, mas também os nomes das rotinas e variáveis.

A inclusão desses símbolos de depuração alarga um aplicativo ou biblioteca significativamente. Aqui estão dois exemplos da quantidade de espaço que esses símbolos ocupam:

- Um binário **bash** com símbolos de depuração: 1200 KB
- Um binário **bash** sem símbolos de depuração: 480 KB (60% menor)
- Arquivos da Glibc e do GCC (`/lib` e `/usr/lib`) com símbolos de depuração: 87 MB
- Arquivos da Glibc e do GCC sem símbolos de depuração: 16 MB (82% menor)

Os tamanhos variarão dependendo de qual compilador e biblioteca C foi usado, porém um programa que tenha sido despojado dos símbolos de depuração usualmente é algo como 50% a 80% menor que o homônimo não despojado dele. Como a maioria dos(as) usantes nunca usará um depurador no software do sistema deles(as), um monte de espaço em disco pode ser recuperado removendo-se esses símbolos. A próxima seção mostra como despojar todos os símbolos de depuração dos programas e das bibliotecas.

8.84. Despojando

Esta seção é opcional. Se o(a) pretendo(a) usante não for um(a) programador(a) e não planeja fazer qualquer depuração do software do sistema, o tamanho do sistema pode ser reduzido em cerca de 2 GB removendo-se os símbolos de depuração, e algumas entradas desnecessárias da tabela de símbolos, de binários e de bibliotecas. Isso não causa nenhum inconveniente real para um(a) usante típico(a) do Linux.

A maioria das pessoas que usa os comandos mencionados abaixo não experiencia quaisquer dificuldades. Entretanto, é fácil cometer um erro e tornar o novo sistema inutilizável. Portanto, antes de executar os comandos **strip**, é uma boa ideia produzir uma cópia de segurança do sistema LFS no estado atual dele.

Um comando **strip** com a opção `--strip-unneeded` remove todos os símbolos de depuração de um binário ou de uma biblioteca. Ele também remove todas as entradas da tabela de símbolos não necessitadas pelo lincador (para bibliotecas estáticas) ou pelo lincador dinâmico (para binários lincados dinamicamente e bibliotecas compartilhadas).

Os símbolos de depuração oriundos de bibliotecas selecionadas são comprimidos com Zstd e preservados em arquivos separados. Essa informação de depuração é necessária para se executar testes de regressão com *valgrind* ou *gdb* posteriormente, no BLFS.

Observe que o **strip** sobrescreverá o arquivo, de binário ou de biblioteca, que ele estiver processando. Isso pode quebrar os processos usando código ou dados oriundos do arquivo. Se o processo executando o **strip** for afetado, o binário, ou a biblioteca, sendo despojado pode ser destruído; isso pode tornar o sistema completamente desusável. Para evitar esse problema, nós copiamos algumas bibliotecas e binários para `/tmp`, despojamos elas lá e as reinstalamos com o comando **install**. (A entrada relacionada em Seção 8.2.1, “Problemas de Atualização” dá a justificativa para usar o comando **install** aqui).



Nota

O nome do carregador de "ELF" é "ld-linux-x86-64.so.2" em sistemas de 64 bits e "ld-linux.so.2" em sistemas de 32 bits. A construção abaixo seleciona o nome correto para a arquitetura atual, excluindo qualquer coisa terminada com `g`, no caso dos comandos abaixo já tiverem sido executados.



Importante

Se existir algum pacote cuja versão seja diferente da versão especificada pelo livro (ou seguindo um aviso de segurança ou satisfazendo preferência pessoal), possivelmente seja necessário atualizar o nome de arquivo da biblioteca em `save_usrlib` ou `online_usrlib`. **Falhar em fazer isso possivelmente torne o sistema completamente desusável.**

```
save_usrlib="$(cd /usr/lib; ls ld-linux*[^g])
             libc.so.6
             libthread_db.so.1
             libquadmath.so.0.0.0
             libstdc++.so.6.0.36
             libitm.so.1.0.0
             libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
  objcopy --only-keep-debug --compress-debug-sections=zstd $LIB $LIB.dbg
  cp $LIB /tmp/$LIB
  strip --strip-unneeded /tmp/$LIB
  objcopy --add-gnu-debuglink=$LIB.dbg /tmp/$LIB
  install -vm755 /tmp/$LIB /usr/lib
  rm /tmp/$LIB
done

online_usrbin="bash find strip"
online_usrlib="libbfd-2.47.20260726.so
              libsframe.so.3.0.0
              libhistory.so.8.3
              libncursesw.so.6.6
              libm.so.6
              libreadline.so.8.3
              libz.so.1.3.2
              libzstd.so.1.5.7
              $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
  cp /usr/bin/$BIN /tmp/$BIN
  strip --strip-unneeded /tmp/$BIN
  install -vm755 /tmp/$BIN /usr/bin
  rm /tmp/$BIN
done

for LIB in $online_usrlib; do
  cp /usr/lib/$LIB /tmp/$LIB
  strip --strip-unneeded /tmp/$LIB
  install -vm755 /tmp/$LIB /usr/lib
  rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name \*.so* ! -name \*.dbg) \
         $(find /usr/lib -type f -name \*.a) \
         $(find /usr/{bin,sbin,libexec} -type f); do
  case "$online_usrbin $online_usrlib $save_usrlib" in
    *$(basename $i)* )
      ;;
    * ) strip --strip-unneeded $i
      ;;
  esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

Um número grande de arquivos será sinalizado como erros, por causa do formato de arquivo deles não é reconhecido. Esses avisos podem ser seguramente ignorados. Eles indicam que aqueles arquivos são scripts, não binários.

8.85. Organizando

Finalmente, limpe alguns arquivos extras deixados pela execução de testes:

```
rm -rf /tmp/{*,.*}
```

Existem também vários arquivos nos diretórios `/usr/lib` e `/usr/libexec` com uma extensão de nome de arquivo de `.la`. Esses são arquivos "libtool archive". Em um sistema moderno Linux, os arquivos `.la` da libtool somente são úteis para a `libltdl`. Nenhuma biblioteca no LFS é esperada ser carregada pela `libltdl` e é sabido que alguns arquivos `.la` conseguem quebrar construções de pacote do BLFS. Remova aqueles arquivos agora:

```
find /usr/lib /usr/libexec -name \*.la -delete
```

Para mais informação acerca de arquivos libtool archive, veja-se a *seção do BLFS "About Libtool Archive (.la) files"*.

O compilador construído em Capítulo 6 e Capítulo 7 ainda está instalado parcialmente e não é mais necessário. Remova-o com:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Finalmente, remova a conta temporária do(a) usante 'tester' criada no início do capítulo anterior.

```
userdel -r tester
```

Capítulo 9. Configuração do Sistema

9.1. Introdução

Este capítulo discute arquivos de configuração e serviços do systemd. Primeiro, os arquivos gerais de configuração necessários para configurar a rede de comunicação são apresentados.

- Seção 9.2, “Configuração Geral da Rede de Comunicação.”
- Seção 9.2.3, “Configurando o nome de dispositivo do sistema.”
- Seção 9.2.4, “Personalizando o Arquivo `/etc/hosts`.”

Segundo, problemas que afetam a configuração adequada de dispositivos são discutidos.

- Seção 9.3, “Visão Geral do Manuseio de Dispositivo e de Módulo.”
- Seção 9.4, “Gerenciando Dispositivos.”

Terceiro, configurar o relógio do sistema e esquema de teclado é mostrado.

- Seção 9.5, “Configurando o Relógio do Sistema.”
- Seção 9.6, “Configurando o Console do Linux.”

Quarto, uma introdução breve aos arquivos de conjuntos de comandos sequenciados e de configuração usados quando o(a) usuário se loga no sistema é apresentado.

- Seção 9.7, “Configurando a Localidade do Sistema.”
- Seção 9.8, “Criando o Arquivo `/etc/inputrc`.”

E finalmente, configurar o comportamento do systemd é discutido.

- Seção 9.10, “Uso e Configuração do Systemd.”

9.2. Configuração Geral da Rede de Comunicação

Esta seção somente se aplica se uma placa de rede de comunicação for para ser configurada.

9.2.1. Arquivos de Configuração da Interface da Rede de Comunicação

Iniciando com a versão 209, o systemd embarca um processo de segundo plano de configuração da rede de comunicação chamado **systemd-networkd** o qual pode ser usado para configuração básica da rede de comunicação. Adicionalmente, desde a versão 213, resolução de nome DNS pode ser lidada por **systemd-resolved** no lugar de um arquivo estático `/etc/resolv.conf`. Ambos os serviços são habilitados por padrão.



Nota

Se você não usará o **systemd-networkd** para configuração da rede de comunicação (por exemplo, quando o sistema não estiver conectado à rede de comunicação ou você desejar usar outro utilitário, como o NetworkManager para configuração da rede de comunicação), [então] desabilite um serviço para evitar uma mensagem de erro durante a inicialização:

```
systemctl disable systemd-networkd-wait-online
```

Arquivos de configuração para "**systemd-networkd**" (e "**systemd-resolved**") podem ser colocados em `/usr/lib/systemd/network` ou `/etc/systemd/network`. Arquivos em `/etc/systemd/network` tem uma prioridade mais alta que aqueles em `/usr/lib/systemd/network`. Existem três tipos de arquivos de configuração: arquivos `.link`, `.netdev` e `.network`. Para descrições detalhadas e conteúdo de exemplo desses arquivos de configuração, consultem-se as páginas de manual "**systemd-link(5)**", "**systemd-netdev(5)**" e "**systemd-network(5)**".

9.2.1.1. Nomeando Dispositivo da Rede de Comunicação

O Udev normalmente atribui nomes de interface da placa de rede de comunicação baseados em características físicas do sistema, tais como `enp2s1`. Se não tiver certeza acerca do que é o nome da sua interface, [então] você sempre pode executar **ip link** depois que tiver inicializado seu sistema.



Nota

Os nomes de interface dependem da implementação e configuração do processo de segundo plano Udev em execução no sistema. O processo de segundo plano Udev para o LFS (**systemd-udevd**, instalado na Seção 8.77, “Systemd-261.2”), não executará, a menos que o sistema LFS seja inicializado. Assim, não é confiável determinar os nomes de interface sendo usados no sistema LFS executando aqueles comandos na distribuição anfitriã, *mesmo que você esteja no ambiente chroot*.

Para a maioria dos sistemas, existe somente uma interface de rede de comunicação para cada tipo de conexão. Por exemplo, o nome clássico de interface para uma conexão com fios é `eth0`. Uma conexão sem fios usualmente terá o nome `wifi0` ou `wlan0`.

Se você preferir usar os nomes clássicos ou personalizados da interface de rede de comunicação, [então] existem três caminhos alternativos para fazer isso:

- Mascara o arquivo `.link` do Udev para a política padrão:

```
ln -s /dev/null /etc/systemd/network/99-default.link
```

- Criar um esquema manual de nomenclatura, por exemplo nomeando as interfaces com alguma coisa como `"internet0"`, `"dmz0"` ou `"lan0"`. Para fazer isso, crie arquivos `.link` em `/etc/systemd/network/` que selecionem um nome explícito ou um esquema melhor de nomenclatura para as tuas interfaces de rede de comunicação. Por exemplo:

```
cat > /etc/systemd/network/10-ether0.link << "EOF"
[Match]
# Muda o endereço MAC conforme apropriado para o seu dispositivo de rede de comunicação
MACAddress=12:34:45:78:90:AB

[Link]
Name=ether0
EOF
```

Veja-se `"systemd.link(5)"` para mais informações.

- Em `/boot/grub/grub.cfg`, passe a opção `"net.ifnames=0"` na linha de comando do núcleo.

9.2.1.2. Configuração de IP Estático

O comando abaixo cria um arquivo básico de configuração para uma configuração de IP Estático (usando ambos `systemd-networkd` e `systemd-resolved`):

```
cat > /etc/systemd/network/10-eth-static.network << "EOF"
[Match]
Name=<nome-dispositivo-rede-de-comunicação>

[Network]
Address=192.168.0.2/24
Gateway=192.168.0.1
DNS=192.168.0.1
Domains=<Seu Nome de Domínio>
EOF
```

Múltiplas entradas de DNS podem ser adicionadas se você tiver mais que um servidor de DNS. Não inclua entradas de DNS ou de Domínios se você pretende usar um arquivo estático `/etc/resolv.conf`.

9.2.1.3. Configuração do DHCP

O comando abaixo cria um arquivo básico de configuração para uma configuração de IPv4 do DHCP:

```
cat > /etc/systemd/network/10-eth-dhcp.network << "EOF"
[Match]
Name=<nome-dispositivo-rede-de-comunicação>

[Network]
DHCP=ipv4

[DHCP]
UseDomains=true
EOF
```

9.2.2. Criando o Arquivo /etc/resolv.conf

Se o sistema será conectado à Internet, [então] ele precisará de alguns meios de resolução de nome do Domain Name Service (DNS) para resolver nomes de domínio da Internet para endereços de IP e vice versa. Isso é melhor alcançado colocando-se o endereço de IP do servidor de DNS, disponível a partir do ISP ou do(a) administrador(a) da rede de comunicação, no /etc/resolv.conf.

9.2.2.1. Configuração do systemd-resolved



Nota

Se usar métodos incompatíveis com systemd-resolved para configurar suas interfaces da rede de comunicação (exemplo: ppp, etc.); ou se usar qualquer tipo de resolvedor local (exemplo: bind, dnsmasq, unbound, etc.); ou qualquer outro aplicativo que gere um /etc/resolv.conf (exemplo: um aplicativo **resolvconf** outro diferente daquele fornecido pelo systemd), [então] o serviço **systemd-resolved** não deveria ser usado.

Para desabilitar systemd-resolved, emita o seguinte comando:

```
systemctl disable systemd-resolved
```

Quando usar o **systemd-resolved** para configuração do DNS, ele cria o arquivo /run/systemd/resolve/stub-resolv.conf. E, se /etc/resolv.conf não existir, [então] ele será criado pelo **systemd-resolved** como um link simbólico para /run/systemd/resolve/stub-resolv.conf. Então, é desnecessário criar um /etc/resolv.conf manualmente.



Nota

Se você quiser usar **systemd-resolved** para o sistema LFS, mas precisar acessar a Internet no ambiente chroot (por exemplo, para construir um pacote BLFS cujo processo de construção exige uma conexão com a Internet), crie o arquivo /etc/resolv.conf seguindo a configuração estática abaixo para o ambiente chroot, de forma que a resolução de nomes funcione no ambiente chroot. Quando você sair do ambiente chroot, remova-o de forma que **systemd-resolved** crie o linque simbólico na inicialização.

9.2.2.2. Configuração Estática do resolv.conf

Se um `/etc/resolv.conf` estático for desejado, [então] crie-o executando o seguinte comando:

```
cat > /etc/resolv.conf << "EOF"
# Início do /etc/resolv.conf

domain <Seu Nome de Domínio>
nameserver <Endereço de IP do seu servidor primário de nome>
nameserver <Endereço de IP do seu servidor secundário de nome>

# Fim do /etc/resolv.conf
EOF
```

A declaração `domain` pode ser omitida ou substituída por uma declaração `search`. Veja-se a página de manual para `resolv.conf` para mais detalhes.

Substitua `<Endereço de IP do servidor de nome>` pelo endereço de IP do servidor DNS mais apropriado para a sua configuração. Frequentemente existirá mais que uma entrada (exigências demandam servidores secundários para capacidade de substituto). Se você precisa ou quer somente um servidor DNS, [então] remova a segunda linha `servidor de nome` do arquivo. O endereço de IP também possivelmente seja um roteador na rede local de comunicação. Outra opção é a de usar o serviço de DNS Google Public usando os endereços de IP abaixo como servidores de nome.



Nota

Os endereços DNS IPv4 do Google Public são `8.8.8.8` e `8.8.4.4` para IPv4; e `2001:4860:4860::8888` e `2001:4860:4860::8844` para IPv6.

9.2.3. Configurando o nome de dispositivo do sistema

Durante o processo de inicialização, o arquivo `/etc/hostname` é usado para estabelecer o nome de dispositivo do sistema.

Crie o arquivo `/etc/hostname` e informe um nome de dispositivo executando:

```
echo "<lfs>" > /etc/hostname
```

`<lfs>` precisa ser substituído pelo nome dado para o computador. Não informe o Fully Qualified Domain Name (FQDN) aqui. Essa informação é colocada no arquivo `/etc/hosts`.

9.2.4. Personalizando o Arquivo `/etc/hosts`

Decida acerca de um "Fully-Qualified Domain Name" ("FQDN"), e possíveis apelidos para uso no arquivo `/etc/hosts`. Se usar endereços estáticos de IP, [então] você também precisará decidir acerca de um endereço de IP. A sintaxe para uma entrada de arquivo "hosts" é:

```
Endereços_IP meuhost.exemplo.org apelidos
```

A menos que o computador seja para estar visível para a Internet (por exemplo, existe um domínio registrado e um bloco válido de endereços atribuídos de IP—a maioria dos(as) usantes não tem isso), assegure-se de que o endereço de IP esteja no intervalo de endereço privado de IP da rede de intercomunicação. Intervalos válidos são:

```
Intervalo de Endereço Privado de Rede Prefixo Normal
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` pode ser qualquer número no intervalo 16-31. `y` pode ser qualquer número no intervalo 0-255.

Um endereço de IP privado válido poderia ser `192.168.1.1`.

Se o computador for para estar visível para a Internet, [então] um "FQDN" válido pode ser o próprio nome de domínio ou uma sequência de caracteres resultante da concatenação de um prefixo (geralmente o nome do dispositivo) e o nome de domínio com um caractere ".". E você precisa contactar o provedor de domínio para resolver o "FQDN" para seu endereço de IP público.

Mesmo se o computador não estiver visível para a Internet, um "FQDN" ainda é necessário para determinados programas, tais como "MTAs", funcionarem corretamente. Um "FQDN" especial, `localhost.localdomain`, pode ser usado para essa finalidade.

Crie o arquivo `/etc/hosts` usando o seguinte comando:

```
cat > /etc/hosts << "EOF"
# Começo /etc/hosts

<192.168.0.2> <FQDN> [apelido1] [apelido2] ...
::1          ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters

# Fim /etc/hosts
EOF
```

Os valores `<192.168.0.2>` e `<FQDN>` precisam ser mudados para usantes ou exigências específicos(as) (se atribuído um endereço de IP por um(a) administrador(a) de rede de intercomunicação/sistema e a máquina estará conectada a uma rede existente de intercomunicação). O(s) nome(s) de apelido(s) opcional(is) pode(m) ser omitido(s) e a linha `<192.168.0.2>` pode ser omitida se você estiver usando uma conexão configurada com DHCP ou autoconfiguração de IPv6 ou usando `localhost.localdomain` como o FQDN.

O `/etc/hostname` não contém entradas para `"localhost"`, `"localhost.localdomain"` ou o nome do dispositivo (sem um domínio) porque eles são manuseados pelo módulo "NSS" `"myhostname"`; leia-se a página de manual `"nss-myhostname(8)"` para detalhes.

A entrada `::1` é o homônimo "IPv6" do 127.0.0.1 e representa a interface de "loopback" "IPv6".

9.3. Visão Geral do Manuseio de Dispositivo e de Módulo

No Capítulo 8, nós instalamos o processo de segundo plano Udev quando systemd foi construído. Antes de entrarmos nos detalhes referentes a como o Udev funciona, um histórico breve dos métodos anteriores de manuseio de dispositivos é oportuno.

Sistemas Linux em geral tradicionalmente usavam um método estático de criação de dispositivo, pelo qual um grande número de nós de dispositivo era criado sob `/dev` (às vezes literalmente milhares de nós), independente de se os dispositivos de hardware correspondentes atualmente existissem. Isso tipicamente era feito via um script **MAKEDEV**, o qual continha um número de chamadas ao aplicativo **mknod** com os números relevantes de dispositivo maior e menor para cada dispositivo possível que pudesse existir no mundo.

Usando o método Udev, nós de dispositivo somente são criados para aqueles dispositivos que são detectados pelo núcleo. Esses nós de dispositivo são criados a cada vez que o sistema inicializa; eles serão armazenados em um sistema de arquivos `devtmpfs` (um sistema de arquivos virtuais que reside inteiramente na memória do sistema). Nós de dispositivo não exigem muito espaço, de forma que a memória que é usada é insignificante.

9.3.1. Histórico

Em fevereiro de 2000, um novo sistema de arquivos chamado `devfs` foi mesclado no núcleo 2.3.46 e foi tornado disponível durante as séries 2.4 de núcleos estáveis. Embora ele estivesse presente no próprio fonte do núcleo, esse método de criar dispositivos dinamicamente nunca recebeu suporte decisivo dos(as) desenvolvedores(as) centrais do núcleo.

O problema principal com a abordagem adotada pelo `devfs` era a maneira como ele lidava com detecção, criação e nomenclatura de dispositivo. O último problema, esse da nomenclatura de nó de dispositivo, era talvez o mais crítico. Geralmente é aceito que, se nomes de dispositivo forem configuráveis, [então] a política de nomenclatura de dispositivo deveria ser escolhida pelos(as) administradores(as) do sistema e não imposta a eles(as) pelos(as) desenvolvedores(as). O sistema de arquivos `devfs` também sofria com algumas condições que eram inerentes ao projeto dele; essas não poderiam ser corrigidas sem uma revisão substancial do núcleo. O `devfs` ficou marcado como obsoleto por um longo tempo, e finalmente foi removido do núcleo em junho de 2006.

Com o desenvolvimento da árvore do núcleo instável 2.5, liberada posteriormente como a série 2.6 dos núcleos estáveis, um novo sistema de arquivos virtuais chamado `sysfs` veio a existir. O trabalho do `sysfs` é o de fornecer informação a respeito da configuração de hardware do sistema para processos do espaço de usante. Com essa representação visível ao espaço de usante, tornou-se possível desenvolver um substituto de espaço de usante para o `devfs`.

9.3.2. Implementação do Udev

9.3.2.1. Sysfs

O sistema de arquivos `sysfs` foi brevemente mencionado acima. Alguém possivelmente questione como o `sysfs` sabe a respeito dos dispositivos presentes em um sistema e quais números de dispositivo deveriam ser usados para eles. Controladores que tenham sido compilados internamente no núcleo registram os objetos deles em `sysfs` (`devtmpfs` internamente) assim que são detectados pelo núcleo. Para controladores compilados como módulos, o registro acontece quando o módulo for carregado. Assim que o sistema de arquivos `sysfs` for montado (em `/sys`), os dados os quais os controladores tenham registrado com `sysfs` ficam disponíveis para os processos de espaço de usante e para o `udev` para processamento (incluindo modificações para nós de dispositivo).

9.3.2.2. Criação de Nó de Dispositivo

Arquivos de dispositivo são criados pelo núcleo no sistema de arquivos `devtmpfs`. Qualquer controlador que deseje registrar um nó de dispositivo usará o `devtmpfs` (via núcleo do controlador) para fazê-lo. Quando uma instância do `devtmpfs` é montada em `/dev`, o nó de dispositivo inicialmente será exposto para o espaço de usante com um nome, permissões e proprietário(a) fixos.

Pouco tempo depois, o núcleo enviará um `uevent` para **udev**. Baseado nas regras especificadas nos arquivos dentro dos diretórios `/etc/udev/rules.d`, `/usr/lib/udev/rules.d` e `/run/udev/rules.d`, **udev** criará linques simbólicos adicionais para o nó de dispositivo ou mudará as permissões, proprietário(a) ou grupo deles, ou modificará a entrada interna (nome) de base de dados do **udev** para aquele objeto.

As regras nesses três diretórios são numeradas e todos os três diretórios são mesclados. Se **udev** não puder encontrar uma regra para o dispositivo que ele esteja criando, [então] ele deixará as permissões e propriedade no que `devtmpfs` usou inicialmente.

9.3.2.3. Carregamento de Módulo

Controladores de dispositivo compilados como módulos possivelmente tenham apelidos construídos dentro deles. Apelidos são visíveis na saída gerada do aplicativo "**modinfo**" e geralmente estão relacionados aos identificadores específicos ao barramento dos dispositivos suportados por um módulo. Por exemplo, o controlador "`snd-fm801`" suporta dispositivos "PCI" com "ID" de fornecedor "0x1319" e "ID" de dispositivo "0x0801" e tem um apelido de "`pci:v00001319d00000801sv*sd*bc04sc01i*`". Para a maioria dos dispositivos, o controlador de barramento exporta o apelido do controlador que lidaria com o dispositivo via "`sysfs`". Por exemplo, o arquivo `/sys/bus/pci/devices/0000:00:0d.0/modalias` pode conter a sequência de caracteres "`pci:v00001319d00000801sv00001319sd00001319bc04sc01i00`". As regras padrão fornecidas com o "Udev" causarão o "**udev**" chamar `/sbin/modprobe` com o conteúdo da variável de ambiente do "`uevent`" "`MODALIAS`" (o qual deveria ser o mesmo que o conteúdo do arquivo "`modalias`" em "`sysfs`"), dessa forma carregando todos os módulos cujos apelidos correspondam a essa sequência de caracteres depois da expansão de carácter curinga.

Nesse exemplo, isso significa que, em adição a *snd-fm801*, o obsoleto (e indesejado) controlador *forte* será carregado se ele estiver disponível. Veja-se abaixo para maneiras nas quais o carregamento de controladores indesejados pode ser evitado.

O próprio núcleo também é capaz de carregar módulos para protocolos de rede de comunicação, sistemas de arquivos e suporte NLS sob demanda.

9.3.2.4. Lidando com Dispositivos Plugáveis a Quente/Dinâmicos

Quando você pluga um dispositivo, como um reproduutor de MP3 Universal Serial Bus (USB), o núcleo reconhece que o dispositivo agora está conectado e gera um uevent. Esse uevent é então tratado pelo **udev** como descrito acima.

9.3.3. Problemas ao Carregar Módulos e Criar Dispositivos

Existem uns poucos possíveis problemas quando se trata de criar automaticamente nós de dispositivos.

9.3.3.1. Um Módulo do Núcleo Não é Carregado Automaticamente

O Udev só carregará um módulo se ele tiver um apelido específico de barramento e o controlador de barramento exportar adequadamente os apelidos necessários para sysfs. Em outros casos, deve-se organizar o carregamento de módulo por outros meios. Com o Linux-7.1.8, o Udev é conhecido por carregar controladores escritos adequadamente para dispositivos INPUT, IDE, PCI, USB, SCSI, SERIO e FireWire.

Para determinar se o controlador de dispositivo que você exige tem o suporte necessário para o Udev, execute **modinfo** com o nome do módulo como o argumento. Agora tente localizar o diretório do dispositivo sob */sys/bus* e verifique se existe um arquivo *modalias* lá.

Se o arquivo *modalias* existir em sysfs, [então] o controlador suporta o dispositivo e pode falar com ele diretamente, mas não tem o apelido, isso é um defeito no controlador. Carregue o controlador sem a ajuda do Udev e espere que o problema seja corrigido posteriormente.

Se não existir arquivo *modalias* no diretório relevante sob */sys/bus*, [então] isso significa que os(as) desenvolvedores(as) do núcleo ainda não adicionaram suporte *modalias* para esse tipo de barramento. Com o Linux-7.1.8, esse é o caso com barramentos ISA. Espere que esse problema seja corrigido em versões posteriores do núcleo.

O Udev não é destinado para carregar controladores “encapsuladores”, tais como *snd-pcm-oss*, e controladores de não hardware, tais como *loop*, de maneira alguma.

9.3.3.2. Um Módulo do Núcleo Não é Carregado Automaticamente e o Udev Não é Destinado para Carregá-lo

Se o módulo “encapsulador” somente aprimora a funcionalidade fornecida por algum outro módulo (por exemplo, *snd-pcm-oss* aprimora a funcionalidade de *snd-pcm* tornando as placas de som disponíveis para aplicações OSS), [então] configure o **modprobe** para carregar o encapsulador depois que o Udev carregar o módulo encapsulado. Para fazer isso, adicione uma linha “softdep” ao arquivo */etc/modprobe.d/<nome_arquivo>.conf* correspondente. Por exemplo:

```
softdep snd-pcm post: snd-pcm-oss
```

Observe que o comando “softdep” também permite dependências “pre:”, ou uma mistura de ambas as dependências “pre:” e “post:”. Veja-se a página de manual “*modprobe.d(5)*” para mais informação a respeito da sintaxe e recursos do “softdep”.

9.3.3.3. O Udev Carrega Algum Módulo Indesejado

Ou não construa o módulo, ou coloque-o na lista negra em um arquivo `/etc/modprobe.d/blacklist.conf` como feito com o módulo *forte* no exemplo abaixo:

```
blacklist forte
```

Módulos em listas negras ainda podem ser carregados manualmente com o comando explícito **modprobe**.

9.3.3.4. O Udev Cria um Dispositivo Incorretamente ou Faz o Link Simbólico Errado

Isso geralmente acontece se uma regra inesperadamente corresponder com um dispositivo. Por exemplo, uma regra mal escrita pode corresponder com ambos um disco SCSI (como desejado) e o dispositivo genérico SCSI correspondente (incorretamente) por fornecedor(a). Encontre a regra infratora e torne-a mais específica, com a ajuda do comando **udevadm info**.

9.3.3.5. Regra do Udev Funciona de Forma Não Confiável

Isso possivelmente seja outra manifestação do problema anterior. Se não, e sua regra usar atributos do `sysfs`, [então] isso possivelmente seja um problema de temporização do núcleo, a ser corrigido em núcleos posteriores. Por hora, você pode contorná-lo criando uma regra que aguarda o atributo usado do `sysfs` e o adiciona ao arquivo `/etc/udev/rules.d/10-wait_for_sysfs.rules` (crie esse arquivo se ele não existir). Por favor, notifique a lista LFS Development se você o fizer e isso ajudar.

9.3.3.6. O Udev Não Cria um Dispositivo

Primeiro, esteja certo(a) de que o driver está construído internamente no núcleo ou já carregado como um módulo e que o Udev não está criando um dispositivo mal nomeado.

Se um controlador do núcleo não exportar os dados dele para o `sysfs`, [então] o Udev carece da informação necessária para criar um nó de dispositivo. Isso é mais provável de acontecer com controladores terceirizados oriundos de fora da árvore do núcleo. Crie um nó de dispositivo estático em `/usr/lib/udev/devices` com os números maior/menor apropriados (veja-se o arquivo `devices.txt` dentro da documentação do núcleo ou a documentação fornecida pelo(a) fornecedor(a) do controlador terceirizado). O nó de dispositivo estático será copiado para `/dev` pelo **udev**.

9.3.3.7. A Ordem de Nomenclatura do Dispositivo Muda Aleatoriamente Depois de Reinicializar

Isso é devido ao fato de o udev, pelo projeto, lidar com `uevents` e carregar módulos em paralelo e, assim, em uma ordem imprevisível. Isso nunca será “corrigido”. Você não deveria confiar nos nomes de dispositivos do núcleo sendo estáveis. Em vez disso, crie suas próprias regras que façam links simbólicos com nomes estáveis baseados em alguns atributos estáveis do dispositivo, tais como um número serial ou a saída gerada dos vários utilitários `*_id` instalados pelo udev. Veja-se a Seção 9.4, “Gerenciando Dispositivos” e Seção 9.2, “Configuração Geral da Rede de Comunicação” para exemplos.

9.3.4. Leitura Útil

Documentação útil adicional está disponível nos seguintes sítios:

- Uma implementação de espaço de usante do `devfs` http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf
- O Sistema de Arquivos `sysfs` <http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf>

9.4. Gerenciando Dispositivos

9.4.1. Lidando com Dispositivos Duplicados

Como explicado na Seção 9.3, “Visão Geral do Manuseio de Dispositivo e de Módulo,” a ordem na qual dispositivos com a mesma função aparecem em `/dev` é essencialmente aleatória. Por exemplo, se você tiver uma câmera web USB e um sintonizador de TV, às vezes `/dev/video0` se referirá à câmera e `/dev/video1` se referirá ao sintonizador; e às vezes depois de uma reinicialização a ordem mudará. Para todas as classes de hardware, exceto placas de som e placas de rede de intercomunicação, isso é corrigível criando-se regras do `udev` para criar links simbólicos persistentes. O caso das placas de rede de intercomunicação está abrangido separadamente na Seção 9.2, “Configuração Geral da Rede de Comunicação,” e configuração de placa de som pode ser encontrado no *BLFS*.

Para cada um dos seus dispositivos que é provável ter esse problema (mesmo que o problema não exista em sua distribuição atual Linux), encontre o diretório correspondente sob `/sys/class` ou `/sys/block`. Para dispositivos de vídeo, isso possivelmente seja `/sys/class/video4linux/videoX`. Descubra os atributos que identificam o dispositivo de maneira única (geralmente, IDs de fornecedor(a) e produto e (ou) números seriais funcionam):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Então escreva regras que criam os links simbólicos, por exemplo:

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Links simbólicos persistentes para webcam e sintonizador
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f", ATTRS{vendor}=="0x109e", SYMLINK+="tv tuner"

EOF
```

O resultado é que os dispositivos `/dev/video0` e `/dev/video1` ainda se referem aleatoriamente ao sintonizador e à câmera web (e, portanto, nunca deveriam ser usados diretamente), mas existem links simbólicos `/dev/tvtuner` e `/dev/webcam` que sempre apontam para o dispositivo correto.

9.5. Configurando o Relógio do Sistema

Esta seção discute como configurar o serviço de sistema **systemd-timedated**, o qual configura o relógio e fuso horário do sistema.

Se você não conseguir se lembrar se ou não o relógio do hardware está configurado para UTC, [então] descubra executando o comando **hwclock --localtime --show**. Isso mostrará o que a hora atual é de acordo com o relógio do hardware. Se essa hora corresponder à que o seu relógio diz, então o relógio do hardware está configurado para hora local. Se a saída gerada originária do **hwclock** não for a hora local, [então] as chances são as de que ele esteja configurado para hora UTC. Verifique isso adicionando ou subtraindo a quantidade apropriada de horas para o fuso horário à (da) hora mostrada pelo **hwclock**. Por exemplo, se você estiver atualmente no fuso horário MST, o qual é conhecido também como GMT -0700, [então] adicione sete horas à hora local.

O **systemd-timedated** lê `/etc/adjtime` e, dependendo do conteúdo do arquivo, configura o relógio para ou UTC ou hora local.

Crie o arquivo `/etc/adjtime` com o seguinte conteúdo se seu relógio do hardware estiver configurado para hora local:

```
cat > /etc/adjtime << "EOF"
0.0 0 0.0
0
LOCAL
EOF
```

Se `/etc/adjtime` não estiver presente na primeira inicialização, [então] **systemd-timedated** assumirá que o relógio do hardware está configurado para UTC e ajustará o arquivo de acordo com isso.

Você também pode usar o utilitário **timedatectl** para dizer ao **systemd-timedated** se o seu relógio do hardware está configurado para UTC ou hora local:

```
timedatectl set-local-rtc 1
```

O **timedatectl** também pode ser usado para mudar a hora e fuso horário do sistema.

Para mudar sua atual hora do sistema, emita:

```
timedatectl set-time YYYY-MM-DD HH:MM:SS
```

O relógio do hardware também será atualizado de acordo.

Para mudar seu fuso horário atual, emita:

```
timedatectl set-timezone FUSO_HORÁRIO
```

Você consegue obter uma lista dos fusos horários disponíveis executando:

```
timedatectl list-timezones
```



Nota

Por favor, observe que o comando **timedatectl** não funciona no ambiente chroot. Ele somente pode ser usado depois que o sistema LFS for inicializado com o systemd.

9.5.1. Sincronização da Hora da Rede de Comunicação

Iniciando com a versão 213, o systemd embarca um processo de segundo plano chamado **systemd-timesyncd** o qual pode ser usado para sincronizar a hora do sistema com servidores remotos do NTP.

O processo de segundo plano não é destinado como um substituto para o bem estabelecido processo de segundo plano do NTP, porém como uma implementação somente cliente do protocolo SNTP, o qual pode ser usado para tarefas menos avançadas e em sistemas de recurso limitado.

Iniciando com o systemd versão 216, o processo de segundo plano **systemd-timesyncd** está habilitado por padrão. Se você desejar desabilitá-lo, [então] emita o seguinte comando:

```
systemctl disable systemd-timesyncd
```

O arquivo `/etc/systemd/timesyncd.conf` pode ser usado para mudar os servidores do NTP que o **systemd-timesyncd** sincroniza com.

Por favor, observe que, quando o relógio do sistema estiver configurado para Hora Local, [então] o **systemd-timesyncd** não atualizará o relógio do hardware.

9.6. Configurando o Console do Linux

Esta seção discute como configurar o serviço de sistema **systemd-vconsole-setup**, o qual configura a fonte do console virtual e mapa de teclas do console.

O serviço **systemd-vconsole-setup** lê o arquivo `/etc/vconsole.conf` para informação de configuração. Decida qual mapa de teclas e fonte de tela será usada. Vários HOWTOs específicos de idioma também podem ajudar com isso, veja-se <https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>. Examine a saída gerada de **localectl list-keymaps** para uma lista dos mapas válidos de teclas do console. Olhe no diretório `/usr/share/consolefonts` para fontes válidas de tela.

O arquivo `"/etc/vconsole.conf"` deveria conter linhas da forma: `VARIÁVEL=valor`. As seguintes variáveis são reconhecidas:

KEYMAP

Essa variável especifica a tabela de mapeamento de tecla para o teclado. Se desconfigurada, [então] ela padroniza para `us`.

KEYMAP_TOGGLE

Essa variável pode ser usada para configurar um segundo mapa de alternância de teclas e é desconfigurada por padrão.

FONT

Essa variável especifica a fonte usada pelo console virtual.

FONT_MAP

Essa variável especifica o mapa de console para ser usado.

FONT_UNIMAP

Essa variável especifica o mapa de fonte Unicode.

Usaremos `C.UTF-8` como localidade para sessões interativas no console do Linux na Seção 9.7, “Configurando a Localidade do Sistema.” As fontes do console fornecidas pelo pacote `Kbd` contendo os glifos para todos os caracteres provenientes das mensagens do aplicativo na localidade `C.UTF-8` são `LatArCyrHeb*.psfu.gz`, `LatGrkCyr*.psfu.gz`, `Lat2-Terminus16.psfu.gz` e `pancyrillic.f16.psfu.gz` em `/usr/share/consolefonts` (as outras fontes de console fornecidas carecem de glifos de alguns caracteres, como as aspas esquerda/direita do Unicode e o Traço inglês do Unicode). Portanto, configure uma delas, por exemplo `Lat2-Terminus16.psfu.gz` como fonte padrão do console:

```
echo FONT=Lat2-Terminus16 > /etc/vconsole.conf
```

Um exemplo para um teclado e console alemão é dado abaixo:

```
cat > /etc/vconsole.conf << "EOF"
KEYMAP=de-latin1
FONT=Lat2-Terminus16
EOF
```

Você consegue mudar o valor de `KEYMAP` em tempo de execução usando o utilitário **localectl**:

```
localectl set-keymap MAPA
```

**Nota**

Por favor, observe que o comando **localectl** não funciona no ambiente `chroot`. Ele somente pode ser usado depois que o sistema LFS for inicializado com o `systemd`.

Você também consegue usar o utilitário **localectl** com os parâmetros correspondentes para mudar o esquema, modelo, variante e opções de teclado do X11:

```
localectl set-x11-keymap ESQUEMA [MODELO] [VARIANTE] [OPÇÕES]
```

Para listar valores possíveis para os parâmetros do **localectl set-x11-keymap**, execute **localectl** com os parâmetros listados abaixo:

`list-x11-keymap-models`

Mostra os modelos conhecidos de mapeamento de teclado do X11.

`list-x11-keymap-layouts`

Mostra esquemas conhecidos de mapeamento de teclado do X11.

`list-x11-keymap-variants`

Mostra variantes conhecidas de mapeamento de teclado do X11.

`list-x11-keymap-options`

Mostra opções conhecidas de mapeamento de teclado do X11.

**Nota**

Usar quaisquer dos parâmetros listados acima exige o pacote `XKeyboard-Config` originário do BLFS.

9.7. Configurando a Localidade do Sistema

Algumas variáveis de ambiente são necessárias para suporte ao idioma nativo. Configurá-las adequadamente resulta em:

- A saída gerada de aplicativos sendo traduzida para seu idioma nativo
- A classificação correta dos caracteres em letras, dígitos e outras classes. Isso é necessário para o **bash** aceitar adequadamente caracteres não ASCII em linhas de comando em localidades não inglesas
- A sequência de ordenação alfabética correta para o país
- O apropriado tamanho padrão de papel
- A formatação correta dos valores monetário, hora e data

Substitua `<ll>` abaixo pelo código de duas letras para teu idioma desejado (por exemplo, `en`) e `<cc>` pelo código de duas letras para o país apropriado (por exemplo, `GB`). `<charmap>` deveria ser substituído pelo mapa de caracteres canônico para a tua localidade escolhida. Modificadores opcionais, tais como `@euro`, também podem estar presentes.

A lista de todas as localidades suportadas pela Glibc pode ser obtida executando-se o seguinte comando:

```
locale -a
```

Mapas de caracteres podem ter um número de apelidos, por exemplo, `ISO-8859-1` também é referenciado como `iso8859-1` e `iso88591`. Alguns aplicativos não conseguem lidar com os vários sinônimos corretamente (por exemplo, exigem que `UTF-8` seja escrito como `UTF-8`, não `utf8`), de forma que é mais seguro, na maioria dos casos, escolher o nome canônico para uma localidade específica. Para determinar o nome canônico, execute o seguinte comando, onde `<nome da localidade>` é a saída gerada dada por **locale -a** para a tua localidade preferida (`en_GB.iso88591` no nosso exemplo).

```
LC_ALL=<nome da localidade> mapa de caracteres da localidade
```

Para a localidade `en_GB.iso88591`, o comando acima imprimirá:

```
ISO-8859-1
```

Isso resulta em uma configuração final de localidade de `en_GB.ISO-8859-1`. É importante que a localidade encontrada usando-se a heurística acima seja testada antes que seja adicionada aos arquivos de iniciação do Bash:

```
LC_ALL=<nome da localidade> locale language
LC_ALL=<nome da localidade> locale charmap
LC_ALL=<nome da localidade> locale int_curr_symbol
LC_ALL=<nome da localidade> locale int_prefix
```

Os comandos acima deveriam imprimir o nome do idioma, a codificação de caracteres usada pela localidade, a moeda local, e o prefixo para discar antes do número de telefone para a finalidade de se alcançar o país. Se quaisquer dos comandos acima falhar com uma mensagem similar àquela mostrada abaixo, isso significa que tua localidade ou não foi instalada no Capítulo 8 ou não é suportada pela instalação padrão da Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Se isso acontecer, você deveria ou instalar a localidade desejada usando o comando **localedef**, ou considerar escolher uma localidade diferente. As instruções posteriores assumem que não existem tais mensagens de erro originárias da Glibc.

Outros pacotes também podem funcionar incorretamente (mas, possivelmente não necessariamente exibam quaisquer mensagens de erro) se o nome da localidade não corresponder às expectativas deles. Nesses casos, investigar-se como outras distribuições do Linux suportam tua localidade poderia fornecer alguma informação útil.

Uma vez que as configurações adequadas de localidade tenham sido determinadas, crie o arquivo `/etc/locale.conf`:

```
cat > /etc/locale.conf << "EOF"
LANG=<ll>_<CC>.<mapa_caracteres><@modificadores>
EOF
```

O aplicativo de shell **/bin/bash** (aqui chamado de “o shell”) usa uma coleção de arquivos de iniciação para ajudar a criar o ambiente para execução. Cada arquivo tem um uso específico e pode afetar o login e os ambientes interativos diferentemente. Os arquivos no diretório `/etc` fornecem configurações globais. Se arquivos equivalentes existirem no diretório inicial, eles poderão substituir as configurações globais.

Um shell interativo de login é iniciado depois de um login exitoso, usando **/bin/login**, lendo o arquivo `/etc/passwd`. Um shell interativo sem login é iniciado na linha de comando (por exemplo, `[prompt]$ /bin/bash`). Um shell não interativo geralmente está presente quando um conjunto de comandos sequenciais de shell está em execução. Não é interativo porque está processando um conjunto de comandos sequenciais e não aguardando por inserção de usuário entre os comandos.

Os shells de login frequentemente não são afetados pelas configurações em `/etc/locale.conf`. Crie o `/etc/profile` para ler as configurações de localidade a partir de `/etc/locale.conf` e exportá-las, mas configure a localidade `c.UTF-8` se executar no console do Linux (para evitar que programas emitam caracteres que o console do Linux não consiga renderizar):

```
cat > /etc/profile << "EOF"
# Inicia /etc/profile

for i in $(locale); do
    unset ${i%=*}
done

if [[ "$TERM" = linux ]]; then
    export LANG=C.UTF-8
else
    source /etc/locale.conf

    for i in $(locale); do
        key=${i%=*}
        if [[ -v $key ]]; then
            export $key
        fi
    done
fi

# Termina /etc/profile
EOF
```

Observe que você consegue modificar o `/etc/locale.conf` com o utilitário **localectl** do systemd. Para usar o **localectl** para o exemplo acima, execute:

```
localectl set-locale LANG="<ll>_<CC>.<mapa_caracteres><@modificadores>"
```

Você também pode especificar outras variáveis de ambiente específicas de idioma, tais como `LANG`; `LC_CTYPE`; `LC_NUMERIC`; ou qualquer outra variável de ambiente oriunda da saída gerada de **locale**. Apenas separe-as com um espaço. Um exemplo onde `LANG` é configurada como `en_US.UTF-8`, porém `LC_CTYPE` é configurada apenas como `en_US` é:

```
localectl set-locale LANG="en_US.UTF-8" LC_CTYPE="en_US"
```



Nota

Por favor, observe que o comando **localectl** não funciona no ambiente chroot. Ele somente pode ser usado depois que o sistema LFS for inicializado com o systemd.

As localidades `c` (padrão) e `en_US` (aquela recomendada para usuários do inglês dos Estados Unidos da América do Norte) são diferentes. `c` usa o conjunto de caracteres de 7 bits US-ASCII e trata bytes com o bit de ordem alta configurado como caracteres inválidos. Esse é o porquê, por exemplo, do comando **ls** substituí-los por pontos de interrogação nessa localidade. Também, uma tentativa de enviar mensagem com tais caracteres a partir do Mutt

ou do Pine resulta em mensagens de não conformidade com RFC sendo enviadas (o conjunto de caracteres na mensagem de saída é indicado como unknown 8-bit). É sugerido que você use a localidade `c` somente se tiver certeza de que nunca precisará de caracteres de 8 bits.

9.8. Criando o Arquivo `/etc/inputrc`

O arquivo `inputrc` é o arquivo de configuração para a biblioteca `readline`, a qual fornece recursos de edição enquanto o(a) usante estiver digitando uma linha a partir do terminal. Ele funciona traduzindo entradas geradas do teclado em ações específicas. `Readline` é usada pelo `Bash` e pela maioria dos outros shells, bem como por muitos outros aplicativos.

A maioria das pessoas não necessita de funcionalidade específica de usante, de modo que o comando abaixo cria um `/etc/inputrc` global usado por qualquer um(a) que se logue. Se posteriormente decidir que precisa sobrepor os padrões em uma base por usante, você pode criar um arquivo `.inputrc` no diretório lar do(a) usante com os mapeamentos modificados.

Para mais informação a respeito de como editar o arquivo `inputrc`, veja-se **info bash** sob a seção *Readline Init File*. **info readline** também é uma boa fonte de informação.

Abaixo está um `inputrc` global genérico junto com comentários para explicar o que as várias opções fazem. Observe que os comentários não podem estar na mesma linha que os comandos. Crie o arquivo usando o seguinte comando:

```
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 8-bit 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

# uncomment for history search mode with up/down
# "\e[A": history-search-backward
# "\e[B": history-search-forward

# End /etc/inputrc
EOF
```

9.9. Criando o Arquivo `/etc/shells`

O arquivo `shells` contém uma lista dos shells de login no sistema. Os aplicativos usam esse arquivo para determinar quando um shell é válido. Para cada shell, uma linha única deveria estar presente, consistindo do caminho do shell relativo à raiz da estrutura de diretórios (`/`).

Por exemplo, esse arquivo é consultado pelo **chsh** para determinar quando um(a) usuário desprivilegiado(a) possa mudar o shell de login para a própria conta dele(a). Se o nome de comando não estiver listado, o(a) usuário terá negada a habilidade de mudar shells.

É uma exigência para aplicativos, tais como GDM, o qual não povoa o navegador de face se ele não puder encontrar `/etc/shells` ou processos de segundo plano do FTP, os quais tradicionalmente proíbem acesso para usantes com shells não incluídos nesse arquivo.

```
cat > /etc/shells << "EOF"
# Início do /etc/shells

/bin/sh
/bin/bash

# Fim do /etc/shells
EOF
```

9.10. Uso e Configuração do Systemd

9.10.1. Configuração Básica

O arquivo `/etc/systemd/system.conf` contém um conjunto de opções para controlar as operações básicas do "systemd". O arquivo padrão tem todas as entradas comentadas com as configurações padrão indicadas. Esse arquivo é onde o nível de registro pode ser mudado, bem como algumas configurações básicas de registro. Veja-se a página de manual `"systemd-system.conf(5)"` para detalhes a respeito de cada opção de configuração.

9.10.2. Desabilitando a Limpeza da Tela em Tempo de Inicialização

O comportamento normal para o systemd é o de limpar a tela ao final da sequência de inicialização. Se desejado, esse comportamento possivelmente seja mudado executando o seguinte comando:

```
mkdir -pv /etc/systemd/system/getty@tty1.service.d

cat > /etc/systemd/system/getty@tty1.service.d/noclear.conf << EOF
[Service]
TTYVTDisallocate=no
EOF
```

As mensagens de inicialização sempre podem ser revistas usando-se o comando `journalctl -b` como o(a) usante root.

9.10.3. Desabilitando tmpfs para /tmp

Por padrão, `/tmp` é criado como um tmpfs. Se isso não for desejado, [então] ele pode ser sobreposto executando-se o seguinte comando:

```
systemctl mask tmp.mount
```

Alternativamente, se uma partição separada para `/tmp` for desejada, [então] especifique essa partição em uma entrada do `/etc/fstab`.

9.10.4. Configurando a Criação e Deleção Automática de Arquivo

Existem vários serviços que criam ou deletam arquivos ou diretórios:

- `systemd-tmpfiles-clean.service`
- `systemd-tmpfiles-setup-dev.service`
- `systemd-tmpfiles-setup.service`

O local de sistema para os arquivos de configuração é `/usr/lib/tmpfiles.d/*.conf`. Os arquivos locais de configuração estão em `/etc/tmpfiles.d`. Os arquivos em `/etc/tmpfiles.d` substituem os arquivos com o mesmo nome em `/usr/lib/tmpfiles.d`. Veja-se a página de manual `"tmpfiles.d(5)"` para detalhes do formato do arquivo.

Observe que a sintaxe para os arquivos `/usr/lib/tmpfiles.d/*.conf` pode ser confusa. Por exemplo, a deleção padrão de arquivos no diretório `/tmp` está localizada em `/usr/lib/tmpfiles.d/tmp.conf` com a linha:

```
q /tmp 1777 root root 10d
```

O campo tipo, `q`, indica a criação de um sub volume com cotas, o qual realmente é aplicável somente para sistemas de arquivos btrfs. Ele referencia tipo `v`, o qual sequencialmente referencia tipo `d` (diretório). Isso então cria o diretório especificado se ele não estiver presente e ajusta as permissões e propriedade como especificado. O conteúdo do diretório estará sujeito a limpeza baseada em hora se o argumento idade for especificado.

Se os parâmetros padrão não forem desejados, então o arquivo deveria ser copiado para `/etc/tmpfiles.d` e editado conforme desejado. Por exemplo:

```
mkdir -p /etc/tmpfiles.d
cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d
```

9.10.5. Sobrepondo o Comportamento Padrão de Serviços

Os parâmetros de uma unidade podem ser sobrepostos criando-se um diretório e um arquivo de configuração em `/etc/systemd/system`. Por exemplo:

```
mkdir -pv /etc/systemd/system/foobar.service.d

cat > /etc/systemd/system/foobar.service.d/foobar.conf << EOF
[Service]
Restart=always
RestartSec=30
EOF
```

Veja-se a página de manual "`systemd.unit(5)`" para mais informação. Depois de criar o arquivo de configuração, execute "`systemctl daemon-reload`" e "`systemctl restart foobar`" para ativar as mudanças para um serviço.

9.10.6. Depurando a Sequência de Inicialização

Em vez de scripts planos de shell usados nos sistemas de inicialização estilo SysVinit ou BSD, o systemd usa um formato unificado para tipos diferentes dos arquivos de inicialização (ou unidades). O comando **systemctl** é usado para habilitar; desabilitar; controlar estado; e obter a situação dos arquivos de unidade. Aqui estão alguns exemplos dos comandos frequentemente usados:

- **systemctl list-units -t <serviço> [--all]**: lista os arquivos carregados de unidade do tipo serviço.
- **systemctl list-units -t <alvo> [--all]**: lista os arquivos carregados de unidade do tipo alvo.
- **systemctl show -p Wants <multi-user.target>**: mostra todas as unidades que dependem do alvo multi-user. Alvos são arquivos especiais de unidade que são análogos a níveis de execução sob o SysVinit.
- **systemctl status <nome_serviço.service>**: mostra a situação do serviço nome_serviço. A extensão `.service` pode ser omitida se não existirem outros arquivos de unidade com o mesmo nome, tais como arquivos `.socket` (os quais criam um soquete de escuta que fornece funcionalidade similar ao `inetd/xinetd`).

9.10.7. Trabalhando com o Diário do Systemd

O registro em um sistema inicializado com o systemd é manuseado com o `systemd-journald` (por padrão), em vez de um processo de segundo plano típico do Unix `syslog`. Você também pode adicionar um processo de segundo plano normal `syslog` e ter ambos operando lado a lado se desejado. O aplicativo `systemd-journald` armazena entradas de diário em um formato binário, em vez de um arquivo plano de registro de texto. Para auxiliar na análise do arquivo, o comando **journalctl** é fornecido. Aqui estão alguns exemplos dos comandos frequentemente usados:

- **journalctl -r**: mostra todo o conteúdo do diário em ordem cronológica reversa.

- **journalctl -u UNIDADE**: mostra as entradas de diário associadas com o arquivo especificado de UNIDADE.
- **journalctl -b[=ID] -r**: mostra as entradas de diário desde a mais recente inicialização bem sucedida (ou para a ID de inicialização) em ordem cronológica reversa.
- **journalctl -f**: fornece funcionalidade similar ao tail -f (seguir).

9.10.8. Trabalhando com Despejos de Núcleo

Despejos de núcleo são úteis para depurar aplicativos quebrados, especialmente quando um processo de aplicativo de segundo plano quebra. Em sistemas inicializados do systemd, o despejamento de núcleo é manuseado pelo **systemd-coredump**. Ele registrará o despejo de núcleo no diário e armazenará o próprio despejo de núcleo em `/var/lib/systemd/coredump`. Para recuperar e processar despejos de núcleo, a ferramenta **coredumpctl** é fornecida. Aqui estão alguns exemplos de comandos frequentemente usados:

- **coredumpctl -r**: lista todos os despejos de núcleo em ordem cronológica reversa.
- **coredumpctl -1 info**: mostra a informação a partir do mais recente despejo de núcleo.
- **coredumpctl -1 debug**: carrega o mais recente despejo de núcleo no *GDB*.

Despejos de núcleo possivelmente usem um monte de espaço em disco. O espaço máximo em disco usado por despejos de núcleo pode ser limitado criando-se um arquivo de configuração em `/etc/systemd/coredump.conf.d`. Por exemplo:

```
mkdir -pv /etc/systemd/coredump.conf.d

cat > /etc/systemd/coredump.conf.d/maxuse.conf << EOF
[Coredump]
MaxUse=5G
EOF
```

Vejam-se as páginas de manual "*systemd-coredump(8)*"; "*coredumpctl(1)*"; e "*coredump.conf.d(5)*" para mais informação.

9.10.9. Processos de Execução Longa

Iniciando com o systemd-230, todos os processos de usante são finalizados quando uma sessão de usante for terminada, mesmo se `nohup` for usado ou o processo usar as funções `daemon()` ou `setsid()`. Isso é uma mudança deliberada de um ambiente historicamente permissivo para um mais restritivo. O novo comportamento possivelmente cause problemas se você depender de programas de execução longa (por exemplo, **screen** ou **tmux**) para continuarem ativos depois de terminar tua sessão de usante. Existem três maneiras de habilitar processos persistentes para continuarem depois que uma sessão de usante for terminada.

- *Habilitar persistência de processo somente para usantes selecionados(as)*: Usantes normais têm permissão para habilitar persistência de processo com o comando **loginctl enable-linger** para os(as) próprios(as) usantes deles(as). Administradores(as) do sistema podem usar o mesmo comando com um argumento *user* ao habilitar para um(a) usante. Esse(a) usante consegue então usar o comando **systemd-run** para iniciar processos de execução longa. Por exemplo: **systemd-run --scope --user /usr/bin/screen**. Se você habilitar a persistência para teu(u) usante, a `user@.service` continuará, mesmo depois que todas as sessões de login forem fechadas e automaticamente iniciará na inicialização do sistema. Isso tem a vantagem de explicitamente permitir e proibir processos para execução depois que a sessão de usante tiver terminado, porém quebra retrocompatibilidade com ferramentas como **nohup** e utilitários que usam `daemon()`.
- *Habilitar persistência de processo abrangente ao sistema*: Você pode configurar `KillUserProcesses=no` em `/etc/systemd/logind.conf` para habilitar a persistência de processo globalmente para todos(as) os(as) usantes. Isso tem o benefício de deixar o método antigo disponível para todos(as) os(as) usantes à custa do controle explícito.

- *Desabilitar em tempo de construção:* Você pode desabilitar a persistência por padrão enquanto construir o systemd adicionando a chave `-D default-kill-user-processes=false` ao comando **meson** para o systemd. Isso desabilita completamente a habilidade do systemd para finalizar processos de usuário ao fim da sessão.

Capítulo 10. Tornando o Sistema LFS Inicializável

10.1. Introdução

É hora de tornar o sistema LFS inicializável. Este capítulo discute a criação do arquivo `/etc/fstab`; construção de um núcleo para o novo sistema LFS; e instalação do carregador de inicialização GRUB, de modo que o sistema LFS possa ser selecionado para iniciar durante a inicialização.

10.2. Criando o Arquivo `/etc/fstab`

O arquivo `/etc/fstab` é usado por alguns aplicativos para determinar onde sistemas de arquivos são para serem montados por padrão; em qual ordem; e quais precisam ser verificados (para erros de integridade) antes da montagem. Crie uma nova tabela de sistemas de arquivos como esta:

```
cat > /etc/fstab << "EOF"
# Início de /etc/fstab

# sistema de arquivos ponto de montagem tipo opções despejo ordem do
# fsck

/dev/<xxx>      /          <fff>      defaults      1          1
/dev/<yyy>      swap       swap       pri=1         0          0

# Fim do /etc/fstab
EOF
```

Substitua "`<xxx>`"; "`<yyy>`"; e "`<fff>`" pelos valores apropriados para o sistema, por exemplo, "`sda2`"; "`sda5`"; e "`ext4`". Para detalhes a respeito dos seis campos nesse arquivo, veja-se "*fstab(5)*".

Sistemas de arquivos com origem MS-DOS ou Windows (isto é, `vfat`, `ntfs`, `smbfs`, `cifs`, `iso9660`, `udf`) precisam de uma opção especial, `utf8`, para a finalidade de caracteres não ASCII nos nomes de arquivo serem interpretados corretamente. Para localidades não UTF-8, o valor de `iocharset` deveria ser configurado para ser o mesmo que o conjunto de caracteres da localidade, ajustado de tal maneira que o núcleo o entenda. Isso funciona se a definição relevante de conjunto de caracteres (encontrada sob File systems -> Native Language Support quando da configuração do núcleo) tiver sido compilada no núcleo ou construída como um módulo. Entretanto, se o conjunto de caracteres da localidade for UTF-8, a correspondente opção `iocharset=utf8` tornaria o sistema de arquivos sensível a maiúsculas e minúsculas. Para corrigir isso, use a opção especial `utf8` em vez de `iocharset=utf8`, para localidades UTF-8. A opção “`codepage`” também é necessária para sistemas de arquivos `vfat` e `smbfs`. Ela deveria ser configurada para o número da página de código usada sob MS-DOS em teu país. Por exemplo, para a finalidade de montar unidades USB flash, um(a) usante do `ru_RU.KOI8-R` precisaria do seguinte na porção de opções da linha `mount` dele(a) em `/etc/fstab`:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

O correspondente fragmento das opções para usantes do `ru_RU.UTF-8` é:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Observe que usar `iocharset` é o padrão para `iso8859-1` (a qual mantém o sistema de arquivos insensível a maiúsculas e minúsculas) e a opção `utf8` diz ao núcleo para converter os nomes de arquivo usando UTF-8, de forma que eles podem ser interpretados no locale UTF-8.

Ao instalar o GRUB com UEFI, a ESP precisa ser formatada como um sistema de arquivos FAT (EXFAT deveria não ser considerado). No núcleo Linux, o controlador VFAT lida com todos os sistemas de arquivo FAT, de modo que esse arquivo conterá `vfat` independentemente. Um exemplo de como você faria acerca de uma entrada para a ESP se pareceria com isto:

```
cat >> /etc/fstab << "EOF"
/dev/<zzz> /boot/efi vfat rw,relatime,codepage=437,iocharset=iso8859-1 0 2
EOF
```

O conjunto de caracteres de E/S `iso8859-1` é usado aqui, pois nós o habilitaremos como uma parte da configuração UEFI do núcleo na Seção 10.3, “Linux-7.1.8.”. Tecnicamente, o conjunto de caracteres de E/S deveria corresponder à tua localidade, conforme nós discutimos acima. No entanto, o nome de todos os arquivos na partição ESP contém somente caracteres ASCII de 7 bits, de modo que tudo funcionará bem, desde que o conjunto de caracteres para tua localidade trate caracteres ASCII de 7 bits da mesma maneira que ISO-8859-1. Por exemplo, UTF-8 é um conjunto de caracteres desse tipo.



Nota

O sistema de arquivos EFI só precisa ser montado ao instalar-se o GRUB. O sistema usa essa partição antes que o núcleo seja carregado e não a usa em outras situações. Uma alternativa a adicionar essa entrada ao arquivo `fstab` é a de montá-lo manualmente antes de executar **grub-install** abaixo na Seção 10.4, “Usando o GRUB para Configurar o Processo de Inicialização.”

É possível também especificar os valores padrão de página de código e `iocharset` para alguns sistemas de arquivos durante a configuração do núcleo. Os parâmetros relevantes são nominados de “Default NLS Option” (`CONFIG_NLS_DEFAULT`); “Default Remote NLS Option” (`CONFIG_SMB_NLS_DEFAULT`); “Default codepage for FAT” (`CONFIG_FAT_DEFAULT_CODEPAGE`); e “Default `iocharset` for FAT” (`CONFIG_FAT_DEFAULT_IOCHARSET`). Não existe maneira de especificar essas configurações para o sistema de arquivos `ntfs` em tempo de compilação do núcleo.

10.3. Linux-7.1.8

O pacote Linux contém o núcleo Linux.

Tempo aproximado de construção: 0,4 - 32 UPC (tipicamente cerca de 2,5 UPC)

Espaço em disco exigido: 1,7 - 14 GB (tipicamente cerca de 2,3 GB)

10.3.1. Instalação do Núcleo

Construir o núcleo envolve uns poucos passos—configuração; compilação; e instalação. Leia-se o arquivo README na árvore do fonte do núcleo para métodos alternativos à maneira que este livro configura o núcleo.



Importante

Construir o núcleo Linux pela primeira vez é uma das tarefas mais desafiadoras no LFS. Acertar depende do hardware específico para o sistema alvo e de suas necessidades específicas. Existem quase 12.000 itens de configuração que estão disponíveis para o núcleo, embora somente cerca de um terço deles sejam necessários para a maioria dos computadores. Os(As) editores(as) do LFS recomendam que os(as) usantes não familiarizados(as) com esse processo sigam os procedimentos abaixo bastante de perto. O objetivo é o de obter um sistema inicial em um ponto onde você consiga se logar na linha de comando quando reinicializar posteriormente na Seção 11.3, “Reinicializando o Sistema.” Nesse ponto, otimização e personalização não é um objetivo.

Para informação geral a respeito da configuração do núcleo, veja-se <https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt>. Informação adicional acerca de configurar e construir o núcleo podem ser encontradas em <https://anduin.linuxfromscratch.org/LFS/kernel-nutshell/>. Essas referências estão um pouco desatualizadas, mas ainda fornecem uma visão geral razoável do processo.

Se tudo mais falhar, você consegue pedir ajuda na lista de discussão *lfs-support*. Observe que a assinatura é exigida para a finalidade de que a lista evite mensagens indesejadas.

Prepare para compilação executando o seguinte comando:

```
make mrproper
```

Isso garante que a árvore do núcleo esteja absolutamente limpa. A equipe do núcleo recomenda que esse comando seja emitido antes de cada compilação do núcleo. Não confie que a árvore do fonte esteja limpa depois de descompactar.

Existem várias maneiras para configurar as opções do núcleo. Usualmente, isso é feito por meio de uma interface orientada por menus, por exemplo:

```
make menuconfig
```

O significado das variáveis opcionais de ambiente do make:

```
LANG=<host_LANG_value> LC_ALL=
```

Isso estabelece a configuração da localidade para aquela usada no anfitrião. Isso possivelmente seja necessário para um adequado desenho de linha da interface ncurses do menuconfig em um console de texto UTF-8 do Linux.

Se usada, assegure-se de substituir *<host_LANG_value>* pelo valor da variável `$LANG` oriunda do teu anfitrião. Você pode, alternativamente, usar, em vez disso, o valor do anfitrião de `$LC_ALL` ou `$LC_CTYPE`.

make menuconfig

Isso lança uma interface ncurses controlada por menu. Para outras (gráficas) interfaces, digite **make help**.

**Nota**

Um bom lugar de partida para configurar a configuração do núcleo é executar **make defconfig**. Isso configurará a configuração base para um bom estado que leve a sua atual arquitetura de sistema em conta.

Assegure-se de habilitar/desabilitar/configurar os seguintes recursos ou o sistema poderia não funcionar corretamente ou inicializar de jeito nenhum:

```

General setup --->
[ ] Compile the kernel with warnings as errors [WERROR]
CPU/Task time and stats accounting --->
[*] Pressure stall information tracking [PSI]
[ ] Require boot parameter to enable pressure stall information tracking
... [PSI_DEFAULT_DISABLED]
< > Enable kernel headers through /sys/kernel/kheaders.tar.xz [IKHEADERS]
[*] Control Group support ---> [CGROUPS]
[*] Memory controller [MEMCG]
[ /*] CPU controller ---> [CGROUP_SCHED]
# This may cause some systemd features malfunction:
[ ] Group scheduling for SCHED_RR/FIFO [RT_GROUP_SCHED]
[ ] Configure standard kernel features (expert users) ---> [EXPERT]

Processor type and features --->
[*] Build a relocatable kernel [RELOCATABLE]
[*] Randomize the address of the kernel image (KASLR) [RANDOMIZE_BASE]

General architecture-dependent options --->
[*] Stack Protector buffer overflow detection [STACKPROTECTOR]
[*] Strong Stack Protector [STACKPROTECTOR_STRONG]

[*] Networking support ---> [NET]
Networking options --->
[*] TCP/IP networking [INET]
[*] The IPv6 protocol ---> [IPV6]

Device Drivers --->
Generic Driver Options --->
[ ] Support for uevent helper [UEVENT_HELPER]
[*] Maintain a devtmpfs filesystem to mount at /dev [DEVTMPFS]
[*] Automount devtmpfs at /dev, after the kernel mounted the rootfs
... [DEVTMPFS_MOUNT]

Firmware loader --->
[ ] Enable the firmware sysfs fallback mechanism [FW_LOADER_USER_HELPER]

Firmware Drivers --->
[*] Export DMI identification via sysfs to userspace [DMIID]
[*] Mark VGA/VBE/EFI FB as generic system framebuffer [SYSFB_SIMPLEFB]

Character devices --->
*- Enable TTY [TTY]
[ ] Allow legacy TIOCSI usage [LEGACY_TIOCSI]

Graphics support --->
<*> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
... [DRM]
[*] Display a user-friendly message when a kernel panic occurs
... [DRM_PANIC]
(kmsg) Panic screen formatter [DRM_PANIC_SCREEN]
Supported DRM clients --->
[*] Enable legacy fbdev support for your modesetting driver
... [DRM_FBDEV_EMULATION]

Drivers for system framebuffers --->
<*> Simple framebuffer driver [DRM_SIMPLEDRM]

Console display driver support --->
[*] Framebuffer Console support [FRAMEBUFFER_CONSOLE]

File systems --->
[*] Inotify support for userspace [INOTIFY_USER]

Pseudo filesystems --->
[*] Tmpfs virtual memory file system support (former shm fs) [TMPFS]
[*] Tmpfs POSIX Access Control Lists [TMPFS_POSIX_ACL]

```

Habilite alguns recursos adicionais se você estiver construindo um sistema de 64 bits. Se você estiver usando o menuconfig, habilite-as na ordem de `CONFIG_PCI_MSI` primeiro; então `CONFIG_IRQ_REMAP`; e finalmente `CONFIG_X86_X2APIC`, pois uma opção somente aparece depois que as dependências dela forem selecionadas.

```
Processor type and features --->
[*] x2APIC interrupt controller architecture support          [X86_X2APIC]

Device Drivers --->
[*] PCI support --->                                         [PCI]
[*] Message Signaled Interrupts (MSI and MSI-X)              [PCI_MSI]
[*] IOMMU Hardware Support --->                             [IOMMU_SUPPORT]
[*] Support for Interrupt Remapping                          [IRQ_REMAP]
```

Se você estiver construindo um sistema de 32 bits, ajuste a configuração, de modo que o núcleo estará apto para usar até 4 GB de RAM física:

```
Processor type and features --->
[*] High Memory Support                                     [HIGHMEM4G]
```

Se a partição para o sistema LFS estiver em um SSD NVME (isto é, o nó do dispositivo para a partição for `/dev/nvme*`, em vez de `/dev/sd*`), habilite o suporte a NVME ou o sistema LFS não inicializaria:

```
Device Drivers --->
NVME Support --->
<*> NVM Express block device                               [BLK_DEV_NVME]
```

Se você estiver inicializando com UEFI, ajuste o núcleo de modo que ele consiga ter suporte de partição e de tempo de execução para EFI no topo do suporte para sistema de arquivos VFAT do DOS, o qual é necessário para a partição EFI:

```
Processor type and features --->
[*] EFI runtime service support                             [EFI]
[*]   EFI stub support                                     [EFI_STUB]

-* Enable the block layer --->                               [BLOCK]
Partition Types --->
[ /*] Advanced partition selection                         [PARTITION_ADVANCED]
[*]   EFI GUID Partition support                          [EFI_PARTITION]

File systems --->
DOS/FAT/EXFAT/NT Filesystems --->
<*/M> VFAT (Windows-95) fs support                        [VFAT_FS]
Pseudo filesystems --->
<*/M> EFI Variable filesystem                             [EFIVAR_FS]
-* Native language support --->                             [NLS]
<*/M> Codepage 437 (United States, Canada)                [NLS_CODEPAGE_437]
<*/M> NLS ISO 8859-1 (Latin 1; Western European Languages) [NLS_ISO8859_1]
```

Se `PARTITION_ADVANCED` não estiver selecionada, `EFI_PARTITION` estará oculta, porém implicitamente selecionada. Não selecione `PARTITION_ADVANCED` apenas porquê você precisa inicializar via UEFI.



Nota

Embora "The IPv6 Protocol" não seja estritamente exigida, ela é altamente recomendada pelos(as) desenvolvedores(as) do systemd.

Existem várias outras opções que possivelmente sejam desejadas, dependendo das exigências para o sistema. Para uma lista das opções necessárias para pacotes do BLFS, veja-se o *Índice do BLFS das Configurações do Núcleo*.

A justificativa para os itens de configuração acima:

Randomize the address of the kernel image (KASLR)

Habilita ASLR para imagem do núcleo, para mitigar alguns ataques baseados em endereços fixos de dados ou código sensíveis no núcleo.

Compile the kernel with warnings as errors

Isso possivelmente cause falha de construção se o compilador e (ou) a configuração forem diferentes daqueles dos(as) desenvolvedores(as) do núcleo.

Enable kernel headers through /sys/kernel/kheaders.tar.xz

Isso exigirá **cpio** ao se construir o núcleo. **cpio** não é instalado pelo LFS.

Configure standard kernel features (expert users)

Isso fará com que algumas opções apareçam na interface de configuração, mas mudar essas opções possivelmente seja perigoso. Não use isso, a menos que você saiba o que está fazendo.

Strong Stack Protector

Habilita SSP para o núcleo. Nós o habilitamos para o espaço inteiro de usuário com `--enable-default-ssp` ao configurar o GCC, porém o núcleo não usa a configuração padrão do GCC para SSP. Nós o habilitamos explicitamente aqui.

Support for uevent helper

Ter essa opção configurada possivelmente interfira com o gerenciamento de dispositivo quando se usar o Udev.

Maintain a devtmpfs

Isso criará nós automatizados de dispositivos, os quais são povoados pelo núcleo, mesmo sem o Udev executando. O Udev então executa no topo disso, gerenciando permissões e adicionando links simbólicos. Esse item de configuração é exigido para todos(as) os(as) usuários do Udev.

Automount devtmpfs at /dev

Isso montará a visão do núcleo dos dispositivos em /dev assim que alternar para o sistema de arquivos raiz pouco antes de iniciar o init.

Display a user-friendly message when a kernel panic occurs

Isso fará com que o núcleo exiba corretamente a mensagem caso ocorra um pânico de núcleo e um controlador DRM em execução suporte fazer isso. Sem isso, seria mais difícil diagnosticar um pânico: se nenhum controlador DRM estiver em execução, nós estaríamos no console VGA que só pode conter 24 linhas e a mensagem relevante do núcleo frequentemente é descartada; se um controlador DRM estiver em execução, a exibição geralmente fica completamente bagunçada no pânico. A partir do Linux-6.12, nenhum dos controladores dedicados para modelos de GPU convencionais realmente suporta isso, mas é suportado pelo “Simple framebuffer driver” que executa no framebuffer VESA (ou EFI) antes do controlador de GPU dedicado ser carregado. Se o controlador de GPU dedicado for construído como um módulo (em vez de parte da imagem do núcleo) e nenhum initramfs for usado, essa funcionalidade funcionará perfeitamente antes que o sistema de arquivos raiz seja montado e já será suficiente para fornecer informações acerca da maioria dos erros de configuração do LFS que causam um pânico (por exemplo, uma configuração `root=` incorreta em Seção 10.4, “Usando o GRUB para Configurar o Processo de Inicialização”).

Panic screen formatter

Configure esse `kmsg` para garantir que as últimas linhas de mensagens do núcleo sejam exibidas quando um pânico de núcleo acontecer. O padrão, `user`, faria o núcleo mostrar somente uma mensagem de pânico “amigável para o(a) usuário”, o que não é útil no diagnóstico. A terceira opção, `qr_code`, faria o núcleo comprimir as últimas linhas de mensagens do núcleo em um código QR e exibi-lo. O código QR pode conter mais linhas de mensagens que texto simples e pode ser decodificado com um dispositivo externo (como um telefone inteligente). Mas exige um compilador Rust que o LFS não fornece.

Mark VGA/VBE/EFI FB as generic system framebuffer e Simple framebuffer driver

Esses permitem usar o framebuffer VESA (ou o framebuffer EFI se inicializar o sistema LFS via UEFI) como um dispositivo DRM. O framebuffer VESA será configurado pelo GRUB (ou o framebuffer EFI será configurado pelo firmware UEFI), de forma que o manuseador de pânico DRM consegue funcionar antes que o controlador DRM específico da GPU seja carregado.

Enable legacy fbdev support for your modesetting driver e Framebuffer Console support

Esses são necessários para exibir o console Linux em uma GPU controlada por um controlador DRI (Direct Rendering Infrastructure). Como `CONFIG_DRM` (Direct Rendering Manager) está habilitada, nós deveríamos habilitar essas duas opções também ou veremos uma tela em branco quando o controlador DRI for carregado.

Support x2apic

Suporta executar o controlador de interrupção dos processadores x86 de 64 bits em modo x2APIC. O x2APIC possivelmente seja habilitado por firmware em sistemas x86 de 64 bits e um núcleo sem essa opção habilitada dará pânico na inicialização se o x2APIC for habilitado por firmware. Essa opção não tem efeito, porém também não danifica, se o x2APIC for desabilitado pelo firmware.

Alternativamente, **make oldconfig** possivelmente seja mais apropriado em algumas situações. Veja-se o arquivo `README` para mais informação.

Se desejado, pule a configuração do núcleo copiando o arquivo de configuração do núcleo, `.config`, a partir do sistema anfitrião (assumindo que ele esteja disponível) para o diretório `linux-7.1.8` desempacotado. Entretanto, nós não recomendamos essa opção. Frequentemente é melhor explorar todos os menus de configuração e criar a configuração do núcleo a partir do zero.

Compile a imagem do núcleo e módulos:

```
make
```

Se usar módulos do núcleo, a configuração do módulo em `/etc/modprobe.d` possivelmente seja exigida. Informação pertinente à configuração de módulos e núcleo está localizada na Seção 9.3, “Visão Geral do Manuseio de Dispositivo e de Módulo” e na documentação do núcleo no diretório `linux-7.1.8/Documentation`. Também, `modprobe.d(5)` possivelmente seja de interesse.

A menos que o suporte a módulo tenha sido desabilitado na configuração do núcleo, instale os módulos com:

```
make modules_install
```

Depois que a compilação do núcleo estiver completa, passos adicionais são exigidos para completar a instalação. Alguns arquivos precisam ser copiados para o diretório `/boot`.



Cuidado

Se você tiver decidido usar uma partição `/boot` separada para o sistema LFS (talvez compartilhando uma partição `/boot` com a distribuição anfitriã), os arquivos copiados abaixo deveriam ir para lá. A maneira mais fácil de fazer isso é a de criar a entrada para `/boot` em `/etc/fstab` primeiro (leia-se a seção anterior para detalhes), então emitir o seguinte comando como o(a) usante `root` no *ambiente chroot*:

```
mount /boot
```

O caminho para o nó de dispositivo está omitido no comando, pois **mount** consegue lê-lo a partir de `/etc/fstab`.

O caminho para a imagem do núcleo possivelmente varie, dependendo da plataforma sendo usada. O nome de arquivo abaixo pode ser mudado para se adequar ao seu gosto, porém o tronco do nome de arquivo deveria ser `vmlinux` para ser compatível com a configuração automática do processo de inicialização descrito na próxima seção. O seguinte comando assume uma arquitetura x86:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinux-7.1.8-lfs-13.1-systemd
```

`system.map` é um arquivo de símbolo para o núcleo. Ele mapeia os pontos de entrada de função de cada função na API do núcleo, bem como os endereços das estruturas de dados do núcleo para o núcleo em execução. Ele é usado como um recurso quando se investigar problemas do núcleo. Emita o seguinte comando para instalar o arquivo de mapa:

```
cp -iv System.map /boot/System.map-7.1.8
```

O arquivo de configuração do núcleo `.config` produzido pelo passo **make menuconfig** acima contém todas as seleções de configuração para o núcleo que foi recém compilado. É uma boa ideia manter esse arquivo para futura referência:

```
cp -iv .config /boot/config-7.1.8
```

Instale a documentação para o núcleo Linux:

```
cp -r Documentation -T /usr/share/doc/linux-7.1.8
```

É importante observar que os arquivos no diretório do fonte do núcleo não são de propriedade do(a) *root*. Sempre que um pacote for descompactado como usante *root* (como nós fizemos dentro do chroot), os arquivos terão os IDs de usante e de grupo do que quer que fossem no computador do(a) empacotador(a). Isso geralmente não é um problema para qualquer outro pacote ser instalado, pois a árvore do fonte será removida depois da instalação. Entretanto, a árvore do fonte do Linux frequentemente é mantida por um longo tempo. Devido a isso, existe uma chance de que qualquer ID de usante que o(a) empacotador(a) usou seja atribuído para alguém na máquina. Essa pessoa então teria acesso de escrita ao fonte do núcleo.



Nota

Em muitos casos, a configuração do núcleo precisará ser atualizada para pacotes que serão instalados posteriormente no BLFS. Diferente de outros pacotes, não é necessário remover a árvore do fonte do núcleo depois que o recém construído núcleo for instalado.

Se a árvore do fonte do núcleo será mantida, execute **chown -R 0:0** no diretório `linux-7.1.8` para assegurar que todos os arquivos sejam de propriedade do(a) usante *root*.

Se você estiver atualizando a configuração e reconstruindo o núcleo a partir de uma árvore retida de fonte do núcleo, normalmente você **não** deveria executar o comando **make mrproper**. O comando expurgaria o arquivo `.config` e todos os arquivos `.o` provenientes da construção anterior. Apesar de ser fácil restaurar `.config` a partir da cópia em `/boot`, expurgar todos os arquivos `.o` ainda é um desperdício: para uma simples mudança de configuração, geralmente somente uns poucos arquivos `.o` precisam ser (re)construídos e o sistema de construção do núcleo ignorará corretamente outros arquivos `.o` se eles não forem expurgados.

Por outro lado, se você tiver atualizado o GCC, você deveria executar **make clean** para expurgar todos os arquivos `.o` provenientes da construção anterior, ou a nova construção possivelmente falhe.



Atenção

Alguma documentação do núcleo recomenda criar um link simbólico a partir de `/usr/src/linux` apontando para o diretório do fonte do núcleo. Isso é específico para núcleos anteriores à série 2.6 e *precisa não* ser criado em um sistema LFS, uma vez que ele pode causar problemas para pacotes que você possivelmente deseje construir tão logo seu sistema base LFS esteja completo.

10.3.2. Conteúdo do Linux

Arquivos instalados: `config-7.1.8`, `vmlinuz-7.1.8-lfs-13.1-systemd` e `System.map-7.1.8`
Diretórios instalados: `/lib/modules` e `/usr/share/doc/linux-7.1.8`

Descrições Curtas

`config-7.1.8`

Contém todas as seleções de configuração para o núcleo

`vmlinuz-7.1.8-lfs-13.1-systemd`

O motor do sistema Linux. Quando se liga o computador, o núcleo é a primeira parte do sistema operacional que se torna carregada. Ele detecta e inicializa todos os componentes do hardware do computador, então torna esses componentes disponíveis como uma árvore de arquivos para o software e transforma uma CPU individual em uma máquina multitarefa capaz de executar dezenas de aplicativos aparentemente ao mesmo tempo

`System.map-7.1.8`

Uma lista de endereços e símbolos; ele mapeia os pontos de entrada e endereços de todas as funções e estruturas de dados no núcleo

10.4. Usando o GRUB para Configurar o Processo de Inicialização

10.4.1. Introdução



Atenção

Configurar o GRUB incorretamente pode tornar teu sistema inoperável sem um dispositivo alternativo de inicialização, como um CD-ROM ou uma unidade USB inicializável. Esta seção não é exigida para inicializar teu sistema LFS. Você possivelmente queira apenas modificar teu carregador de inicialização atual, por exemplo, Grub-Legacy ou GRUB2.

Certifique-se de que um disco de inicialização de emergência esteja pronto para “resgatar” o computador se o computador se tornar desusável (não inicializável). Se já não tiver um dispositivo de inicialização, você consegue criar um. Para a finalidade de que o procedimento abaixo funcione, você precisa saltar adiante para o BLFS e instalar *xorriso* oriundo do pacote *libisoburn*.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. Desativar Inicialização de Segurança

O LFS não tem os pacotes essenciais para suportar a Inicialização Segura. Para configurar o processo de inicialização seguindo as instruções desta seção, a Inicialização Segura precisa ser desativada a partir da interface de configuração do firmware. Leia-se a documentação fornecida pelo(a) fabricante do teu sistema para saber como desativar o suporte de Inicialização Segura.

10.4.3. Convenções de Nomenclatura do GRUB

O GRUB usa estrutura de nomenclatura própria dele para unidades e partições na forma de (h,d,n,m) , onde n é o número da unidade rígida e m é o número da partição. Os números da unidade rígida começam do zero, porém os números da partição começam do um para partições normais (do cinco para partições estendidas). Observe que isso é diferente de versões anteriores, onde ambos os números começavam do zero. Por exemplo, a partição *sda1* é $(hd0,1)$ para o GRUB e *sdb3* é $(hd1,3)$. Em contraste com o Linux, GRUB não considera unidades de CD-ROM como unidades rígidas. Por exemplo, se usar um CD em *hdb* e uma segunda unidade rígida em *hdc*, essa segunda unidade rígida ainda seria $(hd1)$.

10.4.4. Definindo a Configuração

Se inicializar o sistema via BIOS, o GRUB funcionará escrevendo um Código de Carregador de Estágio Inicial no primeiro setor (nominado de Master Boot Record, ou MBR) do disco rígido. Essa área não é parte de nenhum sistema de arquivos. O BIOS carrega e executa o conteúdo do MBR e, em seguida, o Código de Carregador de Estágio Inicial carrega a imagem principal do GRUB a partir da Partição de Inicialização do BIOS. A imagem do GRUB é armazenada como dados brutos em vez de um arquivo (precisa não existir sistema de arquivos na Partição de Inicialização do BIOS), de modo que o Código de Carregador de Estágio Inicial não precise suportar nenhum sistema de arquivos e possa ser pequeno o suficiente para caber no MBR.

Se inicializar o sistema via UEFI, o GRUB funcionará armazenando a imagem principal do GRUB como um arquivo executável PE-COFF em um local padrão na Partição EFI de Sistema: `EFI/BOOT/BOOTX64.EFI` (ou `EFI/BOOT/BOOTIA32.EFI` para `i386-efi`). O firmware UEFI o carrega a partir do local padrão e o executa, iniciando o GRUB.

Muitas funções do GRUB (incluindo inicializar o núcleo Linux) não estão incluídas na imagem principal do GRUB. Em vez disso, elas estão armazenadas em um sistema de arquivos como módulos do GRUB. Esse sistema de arquivos geralmente é montado de maneira que os módulos do GRUB possam ser acessados em `/boot/grub` na maioria das

distribuições Linux. Para evitar o problema do ovo e da galinha, **grub-install** embute os módulos necessários para acessar esse sistema de arquivos na imagem principal do GRUB, de modo que ele consiga encontrar e carregar outros módulos.

O local da partição de inicialização é uma escolha do(a) usante que afeta a configuração. Uma recomendação é a de ter-se uma pequena partição separada (tamanho sugerido de 200 MB) apenas para informações de inicialização. Ao fazer isso, não apenas o LFS, mas qualquer distribuição Linux, poderá acessar os mesmos arquivos de inicialização e, conseqüentemente, qualquer sistema inicializado. Se optar por fazer isso, você precisará montar a partição separada, mover todos os arquivos no diretório atual `/boot` (por exemplo, o núcleo Linux que você recém construiu na seção anterior) para a nova partição. Você precisará, então, desmontar a partição e remontá-la como `/boot`. Se você fizer isso, certifique-se de atualizar o arquivo `/etc/fstab`.



Nota

Se a distribuição anfitriã utilizar uma partição separada para `/boot` e você quiser que o sistema LFS também use essa partição para `/boot`, basta montar a partição em `$LFS/boot` na distribuição anfitriã. O núcleo Linux suporta montar uma partição em múltiplos pontos de montagem.

Deixar `/boot` na partição LFS atual também funcionará, porém a configuração para múltiplos sistemas é mais complicada.

Para exemplos e mais informações acerca de esquemas de partição de inicialização, olhar para a Seção 2.4, “Criando uma Nova Partição” possivelmente seja informativo.

Usando as informações acima, determine o designador apropriado para a partição raiz (ou partição de inicialização, se uma separada for usada). Para o exemplo seguinte, é assumido que a partição raiz (ou inicialização separada) é `sda2`.

As seguintes seções explicam como inicializar com BIOS e UEFI. As instalações do GRUB para BIOS, UEFI de 64 bits e UEFI de 32 bits podem coexistir e compartilhar a mesma configuração. As imagens e os dados ficam em locais diferentes, de modo que você pode criar tanto a Partição de Inicialização do BIOS quanto a Partição EFI de Sistema e instalar o GRUB para todos os tipos suportados de firmware (ou seja, executando três comandos **grub-install**). Se você não tiver certeza acerca do teu tipo de firmware ou se planeja mover o disco rígido para um computador diferente, isso é algo que você pode fazer como uma estratégia abrangente.



Nota

Se você estiver fazendo inicialização UEFI, mas tiver criado a partição de BIOS do Grub, possivelmente seja uma boa ideia executar o comando para BIOS caso a inicialização UEFI não funcione como esperado.



Nota

Se você precisar instalar o GRUB somente para um método de inicialização, não tem que executar comandos para ambos os métodos. Você pode executar apenas o comando para o método de inicialização que precisar.

10.4.4.1. Inicialização com BIOS

Para inicializar com BIOS, certifique-se de que a partição de inicialização esteja montada (se usar uma separada) e de que a Partição de BIOS de inicialização exista. Depois disso, instale os arquivos do GRUB em `/boot/grub` e configure a trilha de inicialização:

**Atenção**

O seguinte comando sobrescreverá o carregador de inicialização atual. Não execute o comando de isso não for desejado, por exemplo, se usar um gerenciador de inicialização de terceiro(a) para gerenciar o MBR.

```
grub-install /dev/sda --target=i386-pc
```

10.4.4.2. Inicializando com UEFI

Para inicializar com UEFI, certifique-se de que a partição de inicialização esteja montada (se usar uma separada) e de que a Partição EFI de Sistema esteja montada em `/boot/efi`. Depois disso, instale os arquivos do GRUB em `/boot/grub` e a imagem principal do GRUB em `/boot/efi/EFI/BOOT/BOOTX64.EFI`:

**Atenção**

O seguinte comando sobrescreverá o arquivo `/boot/efi/EFI/BOOT/BOOTX64.EFI`. Se ele já existir, é provável que seja a entrada de outro carregador de inicialização (por exemplo, a instalação do GRUB proveniente da distribuição anfitriã ou o Gerenciador de Inicialização do Windows). Produza uma cópia de segurança do arquivo, de modo que ele possa ser restaurado posteriormente ou carregado como um carregador secundário de inicialização pela nova instalação do GRUB originária do LFS.

```
grub-install --target=x86_64-efi --removable
```

O comando acima assume que você tenha um firmware UEFI de 64 bits. Se você quiser tornar o sistema inicializável sobre um firmware UEFI de 32 bits, execute o comando com `x86_64-efi` substituído por `i386-efi`.

A opção `--removable` faz com que **grub-install** use o local padrão, `EFI/BOOT/BOOTX64.EFI` (ou `EFI/BOOT/BOOTIA32.EFI` para `i386-efi`), em vez do local que o GRUB prefere (`EFI/GRUB/GRUBX64.EFI` ou `EFI/GRUB/GRUBIA32.EFI`). Usar um local não padrão resultaria no local em uma variável EFI ser registrada, mas o LFS carece do pacote do BLFS `efibootmgr`, o qual é necessário para o GRUB registrar o local na variável EFI.



Nota

Algumas implementações de firmware UEFI, embora raras, ignoram o caminho EFI padrão. Tais sistemas, na maioria das vezes, são antigos, como Lenovo ThinkPads ou desktops/laptops HP. Quando a entrada de inicialização estiver ausente na configuração do firmware, você precisará instalar o pacote do BLFS *efibootmgr* para criar uma entrada de inicialização para UEFI. Se for mais fácil, o pacote pode ser instalado pelo gerenciador de pacotes da distribuição, se aplicável, e usado no anfitrião em vez de no sistema LFS. Isso consegue evitar a necessidade de baixar mais tarballs para o sistema LFS por enquanto.

Primeiro instale o pacote e, em seguida, monte o sistema de arquivos da variável EFI, se ele já não estiver montado:

```
mountpoint /sys/firmware/efi/efivars ||
mount -v -t efivarfs efivarfs /sys/firmware/efi/efivars
```

Agora crie uma entrada de inicialização para a EFI:

```
efibootmgr -c -d /dev/sd<x> \
-p <y> -L "LFS" -1 '\EFI\BOOT\BOOT<X64>.EFI'
```

A unidade `/dev/sd<x>` deveria ser o nó de dispositivo para o disco onde a ESP existir. O número da partição `<y>` deveria corresponder ao número da ESP. Se a ESP estiver em `/dev/sda2`, então o número da partição seria 2. Se você estiver usando UEFI de 32 bits, substitua `<X64>` por `IA32`.

Algum firmware (quebrado) possivelmente exija parâmetros adicionais para **efibootmgr**, como `--full-dev-path` ou `-e 1 -E`. Leia-se a página de manual *efibootmgr(8)* para detalhes.

Agora desmonte o sistema de arquivos da variável EFI:

```
umount -v /sys/firmware/efi/efivars
```

10.4.5. Criando o Arquivo de Configuração do GRUB

Gere o `/boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod part_gpt
insmod ext2

set root=(hd0,2)
set gfxpayload=1024x768x32

menuentry "GNU/Linux, Linux 7.1.8-lfs-13.1-systemd" {
    linux /boot/vmlinuz-7.1.8-lfs-13.1-systemd root=/dev/sda2 ro
}
EOF
```

Os comandos **insmod** carregam os módulos do GRUB nominados `part_gpt` e `ext2`. Apesar da nomenclatura, `ext2` suporta, na verdade, os sistemas de arquivos `ext2`, `ext3` e `ext4`. Em uma configuração típica, os módulos `part_gpt` e `ext2` já estão embutidos na imagem principal do GRUB pelo **grub-install**, e os comandos **insmod** para eles não farão nada. No entanto, eles não danificam de qualquer maneira e podem ser necessários com algumas configurações raras.

O comando **set gfxpayload=1024x768x32** configura a resolução e a profundidade de cor do framebuffer VESA a ser passado para o núcleo. Ele é necessário para o controlador SimpleDRM do núcleo usar o framebuffer VESA. Você pode usar um valor diferente de resolução ou de profundidade de cor que melhor se adeque para teu monitor. Essa linha não faz nada quando o sistema for inicializado via UEFI, mas ela não danifica de qualquer maneira.

**Nota**

A partir da perspectiva do GRUB, os arquivos do núcleo estão relativos à partição usada. Se você usou uma partição `/boot` separada, remova `/boot` da linha *linux* acima. Você também precisará mudar a linha *set root* para apontar para a partição de inicialização.

**Nota**

O designador do GRUB para uma partição possivelmente mude se você adicionou ou removeu alguns discos (incluindo discos removíveis, como dispositivos USB miniatura). A mudança possivelmente cause falha de inicialização, pois o `grub.cfg` se refere a alguns designadores “antigos”. Se desejar evitar tal problema, você possivelmente use o UUID de uma partição e o UUID de um sistema de arquivos, em vez de um designador do GRUB para especificar um dispositivo. Execute **`lsblk -o UUID,PARTUUID,PATH,MOUNTPOINT`** para exibir os UUIDs dos teus sistemas de arquivos (na coluna `UUID`) e partições (na coluna `PARTUUID`). Então substitua `set root=(hdx,y)` por `search --set=root --fs-uuid <UUID do sistema de arquivos onde o núcleo estiver instalado>`; e substitua `root=/dev/sda2` por `root=PARTUUID=<UUID da partição onde o LFS estiver construído>`.

Observe que o UUID de uma partição é completamente diferente do UUID do sistema de arquivos nessa partição. Alguns recursos online possivelmente instruem você a usar o `root=UUID=<UUID do sistema de arquivos>`, em vez do `root=PARTUUID=<UUID da partição>`, porém fazer isso exigirá um `initramfs`, o qual está além do escopo do LFS.

O nome do nó de dispositivo para uma partição em `/dev` também possivelmente mude (muito frequentemente em alguns sistemas com múltiplos discos NVME). Você também pode substituir caminhos para nós de dispositivo, como `/dev/sda1`, por `PARTUUID=<UUID da partição>`, no `/etc/fstab`, para evitar uma potencial falha de inicialização no caso do nome do nó de dispositivo tiver mudado.

O GRUB é um programa extremamente poderoso e ele fornece um tremendo número de opções para inicializar a partir de uma ampla variedade de dispositivos, sistemas operacionais e tipos de partição. Existem também muitas opções para personalização, tais como telas gráficas de abertura; reprodução de sons; entrada gerada de mouse; etc. Os detalhes dessas opções estão além do escopo desta introdução.

**Cuidado**

Existe um comando, `grub-mkconfig`, que consegue escrever um arquivo de configuração automaticamente. Ele usa um conjunto de scripts em `/etc/grub.d/` e destruirá quaisquer personalizações que você fizer. Esses scripts são projetados primariamente para distribuições não fonte e não são recomendados para o LFS. Se você instalar uma distribuição comercial do Linux, existe uma boa chance de que esse programa seja executado. Tenha certeza de produzir uma cópia de segurança do teu arquivo `grub.cfg`.

Capítulo 11. O Fim

11.1. O Fim

Muito bem! O novo sistema LFS está instalado! Nós desejamos a você muito sucesso com seu novo e brilhante sistema Linux construído sob medida.

Possivelmente seja uma boa ideia criar um arquivo `/etc/lfs-release`. Tendo esse arquivo, é muito fácil para você (e para nós se você precisar pedir ajuda em algum ponto) descobrir qual versão do LFS está instalada no sistema. Crie esse arquivo executando:

```
echo 13.1-systemd > /etc/lfs-release
```

Dois arquivos descrevendo o sistema instalado possivelmente sejam usados por pacotes que podem ser instalados no sistema posteriormente, ou em forma de binário ou construindo-os.

O primeiro deles mostra a situação do seu novo sistema com respeito ao Linux Standards Base (LSB). Para criar esse arquivo, execute:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="13.1-systemd"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

O segundo deles contém aproximadamente a mesma informação e é usado pelo systemd e alguns ambientes gráficos de área de trabalho. Para criar esse arquivo, execute:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="13.1-systemd"
ID=lfs
PRETTY_NAME="Linux From Scratch 13.1-systemd"
VERSION_CODENAME="<your name here>"
HOME_URL="https://www.linuxfromscratch.org/lfs/"
RELEASE_TYPE="stable"
EOF
```

Tenha certeza de personalizar os campos 'DISTRIB_CODENAME' e 'VERSION_CODENAME' para tornar o sistema unicamente seu.

11.2. Seja Contado(a)

Agora que terminou o livro, você quer ser contado(a) como um(a) usante do LFS? Vá para <https://www.linuxfromscratch.org/cgi-bin/lfscounter.php> e registre-se como um(a) usante do LFS fornecendo teu nome e a primeira versão do LFS que você tenha usado.

Vamos reinicializar no LFS agora.

11.3. Reinicializando o Sistema

Agora que todo o software foi instalado, é tempo de reinicializar seu computador. Entretanto, ainda existem umas poucas coisas a verificar. Aqui estão algumas sugestões:

- Instale qualquer *firmware* necessário, se o controlador do núcleo para o seu hardware exigir alguns arquivos de firmware para funcionar adequadamente.
- Certifique-se de que uma senha seja configurada para o(a) usante root.
- Uma revisão dos seguintes arquivos de configuração também é apropriada neste ponto.

- /etc/fstab
- /etc/hosts
- /etc/inputrc
- /etc/profile
- /etc/resolv.conf (opcional)
- /etc/vimrc

Agora que nós dissemos isso, vamos em frente para inicializar nossa brilhante e nova instalação do LFS pela primeira vez! *Primeiro saia do ambiente chroot:*

logout

Então desmonte os sistemas virtuais de arquivos:

```
umount -v $LFS/dev/pts
mountpoint -q $LFS/dev/shm && umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Se múltiplas partições foram criadas, [então] desmonte as outras partições antes de desmontar a principal, como isto:

```
umount -v $LFS/home
umount -v $LFS
```

Desmonte o próprio sistema de arquivos do LFS:

```
umount -v $LFS
```

Agora, reinicialize o sistema.

Assumindo que o carregador de inicialização GRUB foi configurado como destacado anteriormente, o menu está configurado para inicializar o *LFS 13.1-systemd* automaticamente.

Quando a reinicialização estiver completa, o sistema LFS estará pronto para uso. O que você verá é um prompt simples “login: ”. Neste ponto, você pode prosseguir para o *Livro BLFS* onde você pode adicionar mais software para atender às suas necessidades.

Se a sua reinicialização **não** for bem-sucedida, é hora de solucionar o problema. Para dicas a respeito de como solucionar problemas iniciais da inicialização, veja-se <https://www.linuxfromscratch.org/lfs/troubleshooting.html>.

11.4. Recursos Adicionais

Obrigado por você ler este livro LFS. Nós esperamos que você tenha achado este livro útil e tenha aprendido mais a respeito do processo de criação do sistema.

Agora que o sistema LFS está instalado, você possivelmente esteja questionando: “E agora?” Para responder a essa pergunta, nós compilamos uma lista de recursos para você.

- Manutenção

Defeitos e avisos de segurança são informados regularmente para todo software. Uma vez que um sistema LFS é compilado a partir do fonte, cabe a você se manter a par de tais informativos. Existem vários recursos online que rastreiam tais informativos, alguns dos quais estão mostrados abaixo:

- *Avisos de Segurança do LFS*

Essa é uma lista de vulnerabilidades de segurança descobertas no livro LFS depois que ele é publicado.

- *Lista de Discussão de Segurança de Código Aberto*

Essa é uma lista de discussão para discussão de falhas de segurança, conceitos e práticas na comunidade do Fonte Aberto.

- Dicas do LFS

As Dicas do LFS são uma coleção de documentos educacionais submetidos por voluntários(as) na comunidade do LFS. As dicas estão disponíveis em <https://www.linuxfromscratch.org/hints/downloads/files/>.

- Listas de discussão

Existem várias listas de discussão do LFS que você possivelmente assine se estiver necessitado(a) de ajuda; quiser se manter atualizado(a) com os mais recentes desenvolvimentos; quiser contribuir para o projeto; e mais. Veja-se Capítulo 1 - Listas de Discussão para mais informação.

- The Linux Documentation Project

O objetivo do The Linux Documentation Project (TLDP) é o de colaborar em todos os problemas de documentação do Linux. O TLDP apresenta uma grande coleção de HOWTOs, guias e páginas de manual. Ele está localizado em <http://www.tldp.org/>.

11.5. Começando Depois do LFS

11.5.1. Decidindo o que fazer a seguir

Agora que o LFS está completo e você tem um sistema inicializável, o que você faz? O próximo passo é o de decidir como usá-lo. Geralmente, existem duas grandes categorias a considerar: estação de trabalho ou servidor. De fato, essas categorias não são mutuamente exclusivas. Os aplicativos necessários para cada categoria podem ser combinados em um sistema, mas vamos vê-los separadamente por enquanto.

Um servidor é a categoria mais simples. Geralmente, isso consiste de um servidor da Web, como o *Servidor HTTP Apache* e um servidor de base de dados, como *MariaDB*. No entanto, outros serviços são possíveis. O sistema operacional embutido em um dispositivo de uso único se enquadra nessa categoria.

Por outro lado, uma estação de trabalho é muito mais complexa. Geralmente ela exige um ambiente gráfico de usante, como *LXDE*, *XFCE*, *KDE* ou *Gnome*, baseado em um *ambiente gráfico* básico e vários aplicativos baseados em gráficos, como o *navegador da web Firefox*, *cliente de correio eletrônico Thunderbird* ou *suíte de escritório LibreOffice*. Esses aplicativos exigem muitos (várias centenas, dependendo dos recursos desejados) mais pacotes de aplicativos e bibliotecas de suporte.

Além do acima, existe um conjunto de aplicativos para gerenciamento de sistemas para todos os tipos de sistemas. Esses aplicativos estão todos no livro BLFS. Nem todos os pacotes são necessários em cada ambiente. Por exemplo *dhcpcd*, normalmente não é apropriado para um servidor e *wireless_tools*, normalmente são úteis somente para um sistema de laptop.

11.5.2. Trabalhando em um ambiente básico do LFS

Quando inicializa inicialmente no LFS, você tem todas as ferramentas internas para construir pacotes adicionais. Infelizmente, o ambiente de usante é bastante esparso. Existem algumas maneiras de melhorar isso:

11.5.2.1. Trabalhar a partir do anfitrião LFS em chroot

Esse método fornece um ambiente gráfico completo onde um navegador completo e recursos de copiar/colar estão disponíveis. Esse método permite usar aplicativos, como a versão do anfitrião do Wget, para baixar os fontes do pacote para um local disponível ao se trabalhar no ambiente "chroot".

Para a finalidade de construir adequadamente pacotes no chroot, você também precisará se lembrar de montar os sistemas virtuais de arquivos, se eles ainda não estiverem montados. Uma maneira de fazer isso é a de criar um script no sistema **ANFITRIÃO**:

```
cat > ~/mount-virt.sh << "EOF"
#!/bin/bash

function mountbind
{
    if ! mountpoint $LFS/$1 >/dev/null; then
        $SUDO mount --bind /$1 $LFS/$1
        echo $LFS/$1 mounted
    else
        echo $LFS/$1 already mounted
    fi
}

function mounttype
{
    if ! mountpoint $LFS/$1 >/dev/null; then
        $SUDO mount -t $2 $3 $4 $5 $LFS/$1
        echo $LFS/$1 mounted
    else
        echo $LFS/$1 already mounted
    fi
}

if [ $EUID -ne 0 ]; then
    SUDO=sudo
else
    SUDO=""
fi

if [ x$LFS == x ]; then
    echo "LFS not set"
    exit 1
fi

mountbind dev
mounttype dev/pts devpts devpts -o gid=5,mode=620
mounttype proc    proc    proc
mounttype sys     sysfs   sysfs
mounttype run     tmpfs    run
if [ -h $LFS/dev/shm ]; then
    install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
    mounttype dev/shm tmpfs tmpfs -o nosuid,nodev
fi

#mountbind usr/src
#mountbind boot
#mountbind home
EOF
```

Observe que os últimos três comandos no script são comentados. Eles são úteis se aqueles diretórios forem montados como partições separadas no sistema anfitrião e serão montados quando inicializar o sistema LFS/BLFS completo.

O conjunto de comandos sequenciados pode ser executado com **bash ~/mount-virt.sh** como ou um(a) usante comum (recomendado) ou como root. Se executado como um(a) usante comum, sudo é exigido no sistema anfitrião.

Outro problema apontado pelo conjunto de comandos sequenciados é o de onde armazenar arquivos baixados do pacote. Esse local é arbitrário. Ele pode estar em um diretório lar de usante comum, como ~/sources ou em um local global, como /usr/src. Nossa recomendação é a de não misturar fontes BLFS e fontes LFS em (a partir do ambiente chroot) /sources. Em qualquer caso, os pacotes precisam estar acessíveis dentro do ambiente chroot.

Um último recurso de conveniência apresentado aqui é o de simplificar o processo de entrada no ambiente chroot. Isso pode ser feito com um apelido colocado em um arquivo `~/.bashrc` de usante no sistema anfitrião:

```
alias lfs='sudo /usr/sbin/chroot /mnt/lfs /usr/bin/env -i HOME=/root TERM="$TERM" PS1="\u:\w\\\$ "
PATH=/usr/bin:/usr/sbin /bin/bash --login'
```

Esse apelido é um pouco complicado, por causa das aspas e dos níveis dos caracteres de barra invertida. Precisa estar tudo em uma linha. O comando acima foi dividido em dois para propósitos de apresentação.

11.5.2.2. Trabalhar remotamente via ssh

Esse método também fornece um ambiente completo gráfico, mas primeiro exige a instalação de *sshd* no sistema LFS, geralmente em "chroot". Também exige um segundo computador. Esse método tem a vantagem de ser simples, por não exigir a complexidade do ambiente "chroot". Ele também usa seu núcleo do LFS construído para todos os pacotes adicionais e ainda fornece um sistema completo para instalar pacotes.

Você possivelmente use o comando **scp** para carregar os fontes dos pacotes a serem construídos no sistema LFS. Se você quiser baixar os fontes diretamente no sistema LFS, [então] instale *libtasn1*, *p11-kit*, *make-ca* e *wget* em "chroot" (ou carregue os fontes deles usando **scp** depois de inicializar o sistema LFS).

11.5.2.3. Trabalhar a partir da linha de comando do LFS

Esse método exige instalar *libtasn1*, *p11-kit*, *make-ca*, *wget*, *gpm* e *links* (ou *lynx*) no chroot e, em seguida, reinicializar no novo sistema LFS. Neste ponto, o sistema padrão tem seis consoles virtuais. Alternar consoles é tão fácil quanto usar as combinações de teclas **Alt+Fx**, onde **Fx** está entre **F1** e **F6**. As combinações **Alt+←** e **Alt+→** também mudarão o console.

Neste ponto, você pode logar-se em dois consoles virtuais e executar o navegador Links ou o Lynx em um console e o Bash no outro. O GPM então permite copiar comandos a partir do navegador com o botão esquerdo do mouse, alternar consoles e colar no outro console.



Nota

Como uma observação lateral, alternância de consoles virtuais também pode ser feita a partir de uma instância do X Window com a combinação de teclas **Ctrl+Alt+Fx**, mas a operação de cópia do mouse não funciona entre a interface gráfica e um console virtual. Você pode retornar para exibição do X Window com a combinação **Ctrl+Alt+Fx**, onde **Fx** geralmente é **F1**, mas possivelmente seja **F7**.

Parte V. Anexos

Apêndice A. Siglas e Termos

ABI	Application Binary Interface
ALFS	Automated Linux From Scratch
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input/Output System
BLFS	Beyond Linux From Scratch
BSD	Berkeley Software Distribution
chroot	change root
CMOS	Complementary Metal Oxide Semiconductor
COS	Class Of Service
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CVS	Concurrent Versions System
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
EGA	Enhanced Graphics Adapter
ELF	Executable and Linkable Format
EOF	End of File
EQN	equation
ext2	sistema de arquivos segundo estendido
ext3	sistema de arquivos terceiro estendido
ext4	sistema de arquivos quarto estendido
FAQ	Frequently Asked Questions
FHS	Filesystem Hierarchy Standard
FIFO	First-In, First Out
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GB	Gigabytes
GCC	GNU Compiler Collection
GID	Group Identifier
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronic Engineers
IO	Input/Output
IP	Internet Protocol
IPC	Inter-Process Communication

IRC	Internet Relay Chat
ISO	International Organization for Standardization
ISP	Internet Service Provider
KB	Kilobytes
LED	Light Emitting Diode
LFS	Linux From Scratch
LSB	Linux Standard Base
MB	Megabytes
MBR	Master Boot Record
MD5	Message Digest 5
NIC	Network Interface Card
NLS	Native Language Support
NNTP	Network News Transport Protocol
NPTL	Native POSIX Threading Library
OSS	Open Sound System
PCH	Pre-Compiled Headers
PCRE	Perl Compatible Regular Expression
PID	Process Identifier
PTY	pseudo terminal
QOS	Quality Of Service
RAM	Random Access Memory
RPC	Remote Procedure Call
RTC	Real Time Clock
SBU	Standard Build Unit
SCO	The Santa Cruz Operation
SHA1	Secure-Hash Algorithm 1
TLDP	The Linux Documentation Project
TFTP	Trivial File Transfer Protocol
TLS	Thread-Local Storage
UID	User Identifier
umask	máscara de usante de criação de arquivo
USB	Universal Serial Bus
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
VC	Virtual Console
VGA	Video Graphics Array
VT	Virtual Terminal

Apêndice B. Reconhecimentos

Nós gostaríamos de agradecer às seguintes pessoas e organizações pelas contribuições delas para o Linux From Scratch Project.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Criador do LFS
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Editor-chefe do LFS
- *Jim Gifford* <jim@linuxfromscratch.org> – Colíder do Projeto CLFS
- *Pierre Labastie* <pierre@linuxfromscratch.org> – Editor do BLFS e Líder do ALFS
- *DJ Lucas* <dj@linuxfromscratch.org> – Editor do LFS e BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Editor do BLFS
- Incontáveis outras pessoas nas várias listas de discussão do LFS e do BLFS que ajudaram a tornar este livro possível dando as sugestões delas; testando o livro; e submetendo relatórios de defeitos; instruções; e suas experiências com a instalação de vários pacotes.

Tradutores(as)

- *Manuel Canales Esparcia* <macana@macana-es.com> – Projeto de tradução do LFS para espanhol
- *Johan Lenglet* <johan@linuxfromscratch.org> – Projeto de tradução do LFS para francês até 2008
- *Jean-Philippe Mengual* <jmengual@linuxfromscratch.org> – projeto francês de tradução do LFS 2008-2016
- *Julien Lepiller* <jlepillier@linuxfromscratch.org> – projeto francês de tradução do LFS 2017-present
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Histórico do projeto de tradução LFS para o português
- *Jamenson Espindula* <jafesp@gmail.com> – Projeto de tradução do LFS para o português 2022-presente
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – projeto alemão de tradução do LFS

Mantenedores(as) de Espelhos

Espelhos da América do Norte

- *Scott Kveton* <scott@osuosl.org> – espelho lfs.oregonstate.edu
- *William Astle* <lost@l-w.net> – espelho ca.linuxfromscratch.org
- *Eujon Sellers* <jpolen@rackspace.com> – espelho lfs.introspeed.com
- *Justin Knierim* <tim@idge.net> – espelho lfs-matrix.net

Espelhos da América do Sul

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – espelho lfsmirror.lfs-es.info
- *Luis Falcon* <Luis Falcon> – espelho torredehanoi.org

Espelhos Europeus

- *Guido Passet* <guido@primerelay.net> – espelho nl.linuxfromscratch.org
- *Bastiaa Jacques* <baafie@planet.nl> – espelho lfs.pagefault.net
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – espelho lfs.lineo.be
- *Scarlet Belgium* – espelho lfs.scarlet.be
- *Sebastian Faulborn* <info@aliensoft.org> – espelho lfs.aliensoft.org

- *Stuart Fox* <stuart@dontuse.ms> – espelho lfs.dontuse.ms
- *Ralf Uhlemann* <admin@realhost.de> – espelho lfs.oss-mirror.org
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – espelho at.linuxfromscratch.org
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – espelho se.linuxfromscratch.org
- *Franck* <franck@linuxpourtout.com> – espelho lfs.linuxpourtout.com
- *Philippe Baque* <baque@cict.fr> – espelho lfs.cict.fr
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – espelho lfs.pilgrims.ru
- *Benjamin Heil* <kontakt@wankoo.org> – espelho lfs.wankoo.org
- *Anton Maisak* <info@linuxfromscratch.org.ru> – espelho linuxfromscratch.org.ru

Espelhos Asiáticos

- *Satit Phernsawang* <satit@wbac.ac.th> – espelho lfs.phayoune.org
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – espelho lfs.mirror.shizu-net.jp

Espelhos da Austrália

- *Jason Andrade* <jason@dstc.edu.au> – espelho au.linuxfromscratch.org

Ex-membros da Equipe do Projeto

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – Editor do Livro LFS
- *Archaic* <archaic@linuxfromscratch.org> – Escritor/Editor Técnico do LFS (Dicas e Patches); Líder do Projeto HLFS; Editor do BLFS; Mantenedor do Projeto Dicas e Patches
- *Matthew Burgess* <matthew@linuxfromscratch.org> – Líder de Projeto do LFS; Escritor/Editor Técnico do LFS
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Mantenedor de Scripts de Inicialização do LFS
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Desenvolvedor de Sítio da Web; Mantenedor de FAQ
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Mantenedor de XML e XSL do LFS/BLFS/HLFS
- Alex Groenewoud – Escritor Técnico do LFS
- Marc Heerdink
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – Escritor Técnico do LFS; Mantenedor de LiveCD do LFS
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – Escritor Técnico do LFS
- Mark Hymers
- Seth W. Klein – Mantenedor do FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Mantenedor da Wiki
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Mantenedor de Scripts de Infraestrutura de Sítio Web
- *Anderson Lizardo* <dj@lucasit.com> – Escritor Técnico do LFS
- *Randy McMurphy* <randy@linuxfromscratch.org> – Líder de Projeto do BLFS; Editor do LFS

- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – Editor do LFS e BLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – Escritor Técnico do LFS; Editor de Internacionalização do LFS; Mantenedor de Live CD do LFS
- *Simon Perreault*
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Mantenedor do Gateway NNTP do LFS
- *Douglas R. Reno* <renodr@linuxfromscratch.org> – Editor do Systemd
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Colíder de Projeto do CLFS
- *Greg Schafer* <gschafer@zip.com.au> – Escritor Técnico do LFS e Arquiteto do Método de Construção de Habilitação de 64 bits de Próxima Geração
- *Jesse Tie-Ten-Quee* – Escritor Técnico do LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – Mantenedor do Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Editor do Livro BLFS; Líder de Projeto de Dicas e Patches
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Escritor Técnico do LFS; Mantenedor do Bugzilla; Mantenedor de Scripts de Inicialização do LFS
- *Zack Winkles* <zwinkles@gmail.com> – Escritor Técnico do LFS

Apêndice C. Dependências

Cada pacote construído no LFS depende de um ou mais outros pacotes para a finalidade de construir e instalar adequadamente. Alguns pacotes até participam em dependências circulares, isto é, o primeiro pacote depende do segundo o qual, na sequência, depende do primeiro. Por causa dessas dependências, a ordem na qual pacotes são construídos no LFS é muito importante. O propósito desta página é o de documentar as dependências de cada pacote construído no LFS.

Para cada pacote que é construído, existem três, e às vezes até cinco tipos de dependências listadas abaixo. A primeira lista que outros pacotes necessitam estar disponíveis para a finalidade de compilar e instalar o pacote em questão. A segunda lista os pacotes que precisam estar disponíveis quando quaisquer aplicativos ou bibliotecas oriundos do pacote forem usados em tempo de execução. A terceira lista que pacotes, em adição àqueles na primeira lista, necessitam estar disponíveis para a finalidade de executar as suítes de teste. A quarta lista de dependências são pacotes que exigem que esse pacote esteja construído e instalado no local final dele antes que eles sejam construídos e instalados.

A última lista de dependências são pacotes opcionais que não são endereçados no LFS, porém poderiam ser úteis para o(a) usante. Esses pacotes possivelmente tenham dependências adicionais obrigatórias ou opcionais deles próprios. Para essas dependências, a prática recomendada é a de instalá-las depois de completar o livro LFS e então voltar e reconstruir o pacote LFS. Em muitos casos, a reinstalação está endereçada no BLFS.

Acl

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Automake, Diffutils, Findutils e Libtool
Precisa ser instalado antes de:	Coreutils, Sed, Tar e Vim
Dependências opcionais:	Nenhum

Attr

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Automake, Diffutils, Findutils e Libtool
Precisa ser instalado antes de:	Acl, Libcap e Patch
Dependências opcionais:	Nenhum

Autoconf

A Instalação depende de:	Bash, Coreutils, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução:	Bash, Coreutils, Grep, M4, Make, Sed e Texinfo
A suíte de teste depende de:	Automake, Diffutils, Findutils, GCC e Libtool
Precisa ser instalado antes de:	Automake e Coreutils
Dependências opcionais:	<i>Emacs</i>

Automake

A Instalação depende de:	Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução:	Bash, Coreutils, Grep, M4, Sed e Texinfo
A suíte de teste depende de:	Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool e Tar
Precisa ser instalado antes de:	Coreutils
Dependências opcionais:	Nenhum

Bash

A Instalação depende de:	Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed e Texinfo
Exigido em tempo de execução:	Glibc, Ncurses e Readline
A suíte de teste depende de:	Expect e Shadow
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Xorg</i>

Bc

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make e Readline
Exigido em tempo de execução:	Glibc, Ncurses e Readline
A suíte de teste depende de:	Gawk
Precisa ser instalado antes de:	Linux
Dependências opcionais:	Nenhum

Binutils

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, File, Flex, Gawk, GCC, Glibc, Grep, Make, Perl, Pkgconf, Sed, Texinfo, Zlib e Zstd
Exigido em tempo de execução:	Glibc, Zlib e Zstd
A suíte de teste depende de:	DejaGNU e Expect
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Elfutils</i> e <i>Jansson</i>

Bison

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Diffutils, Findutils e Flex
Precisa ser instalado antes de:	Kbd e Tar
Dependências opcionais:	<i>Doxygen</i>

Bzip2

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make e Patch
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	File e Libelf
Dependências opcionais:	Nenhum

Coreutils

A Instalação depende de:	Autoconf, Automake, Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, OpenSSL, Patch, Perl, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Diffutils, E2fsprogs, Findutils, Shadow e Util-linux
Precisa ser instalado antes de:	Bash, Diffutils, Findutils, Man-DB e Systemd
Dependências opcionais:	<i>Expect.pm</i> e <i>IO::Tty</i>

D-Bus

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Pkgconf, Sed, Systemd e Util-linux
Exigido em tempo de execução:	Glibc e Systemd
A suíte de teste depende de:	Vários pacotes no BLFS
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Bibliotecas do Xorg</i>

DejaGNU

A Instalação depende de:	Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Expect e Bash
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Diffutils

A Instalação depende de:	Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Perl
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

E2fsprogs

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Pkgconf, Sed, Systemd, Texinfo e Util-linux
Exigido em tempo de execução:	Glibc e Util-linux
A suíte de teste depende de:	Procps-ng e Psmisc
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Expat

A Instalação depende de:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Python e XML::Parser
Dependências opcionais:	Nenhum

Expect

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed e Tcl
Exigido em tempo de execução:	Glibc e Tcl
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Tk</i>

File

A Instalação depende de:	Bash, Binutils, Bzip2, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, Xz, Zlib e Zstd
Exigido em tempo de execução:	Glibc, Bzip2, Xz e Zlib
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>libseccomp</i>

Findutils

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Bash e Glibc
A suíte de teste depende de:	DejaGNU, Diffutils e Expect
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Flex

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed e Texinfo
Exigido em tempo de execução:	Bash, Glibc e M4
A suíte de teste depende de:	Bison e Gawk
Precisa ser instalado antes de:	Binutils, IProute2, Kbd, Kmod e Man-DB
Dependências opcionais:	Nenhum

Flit-Core

A Instalação depende de:	Python
Exigido em tempo de execução:	Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Packaging e Wheel
Dependências opcionais:	<i>pytest</i> e <i>testpath</i>

Gawk

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, MPFR, Patch, Readline, Sed e Texinfo
Exigido em tempo de execução:	Bash, Glibc e Mpfr
A suíte de teste depende de:	Diffutils
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>libsigsegv</i>

GCC

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, Texinfo e Zstd
Exigido em tempo de execução:	Bash, Binutils, Glibc, Mpc e Python
A suíte de teste depende de:	DejaGNU, Expect e Shadow
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>GDC, GNAT e ISL</i>

GDBM

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make e Sed
Exigido em tempo de execução:	Bash, Glibc e Readline
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Gettext

A Instalação depende de:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Sed e Texinfo
Exigido em tempo de execução:	Acl, Bash, Gcc e Glibc
A suíte de teste depende de:	Diffutils, Perl e Tcl
Precisa ser instalado antes de:	Automake e Bison
Dependências opcionais:	<i>libunistring</i> e <i>libxml2</i>

Glibc

A Instalação depende de:	Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Cabeçalhos da API do Linux, Make, Perl, Python, Sed e Texinfo
Exigido em tempo de execução:	Nenhum
A suíte de teste depende de:	File
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

GMP

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed e Texinfo
Exigido em tempo de execução:	GCC e Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	MPFR e GCC
Dependências opcionais:	Nenhum

Gperf

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc e Make
Exigido em tempo de execução:	GCC e Glibc
A suíte de teste depende de:	Diffutils e Expect
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Grep

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Pcre2, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Gawk
Precisa ser instalado antes de:	Man-DB
Dependências opcionais:	Nenhum

Groff

A Instalação depende de:	Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed e Texinfo
Exigido em tempo de execução:	GCC, Glibc e Perl
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Man-DB
Dependências opcionais:	<i>ghostscript</i> e <i>Uchardet</i>

GRUB

A Instalação depende de:	Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo e Xz
Exigido em tempo de execução:	Bash, GCC, Gettext, Glibc, Xz e Sed
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Gzip

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Bash e Glibc
A suíte de teste depende de:	Diffutils e Less
Precisa ser instalado antes de:	Man-DB
Dependências opcionais:	Nenhum

lana-Etc

A Instalação depende de:	Coreutils
Exigido em tempo de execução:	Nenhum
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Perl
Dependências opcionais:	Nenhum

Inetutils

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo e Zlib
Exigido em tempo de execução:	GCC, Glibc, Ncurses e Readline
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Tar
Dependências opcionais:	Nenhum
Exigido em tempo de execução:	Autoconf, Automake, Bash, Glibc, Grep, Perl e Sed
A suíte de teste depende de:	Perl
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

IProute2

A Instalação depende de:	Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf, Cabeçalhos da API do Linux, Pkgconf e Zlib
Exigido em tempo de execução:	Bash, Coreutils, Glibc, Libcap, Libelf e Zlib
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Berkeley DB, iptables, libbbpf, libmnl e libtirpc</i>

Jinja2

A Instalação depende de:	MarkupSafe, Python, Setuptools e Wheel
Exigido em tempo de execução:	MarkupSafe e Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Systemd
Dependências opcionais:	Nenhum

Kbd

A Instalação depende de:	Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch e Sed
Exigido em tempo de execução:	Bash, Coreutils e Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Linux-PAM</i>

Kmod

A Instalação depende de:	Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkgconf, Sed, Xz, Zlib e Zstd
Exigido em tempo de execução:	Glibc, Xz e Zlib
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Systemd
Dependências opcionais:	<i>scdoc</i> (para páginas de manual)

Less

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Pcre2 e Sed
Exigido em tempo de execução:	Glibc e Ncurses
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Gzip
Dependências opcionais:	Nenhum

Libcap

A Instalação depende de:	Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	IProute2 e Shadow
Dependências opcionais:	<i>Linux-PAM</i>

Libelf

A Instalação depende de:	Bash, Binutils, Bzip2, Coreutils, GCC, Glibc, Make, Xz, Zlib e Zstd
Exigido em tempo de execução:	Bzip2, Glibc, Xz, Zlib e Zstd
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	IProute2 e Linux
Dependências opcionais:	Nenhum

Libffi

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Make e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	DejaGnu
Precisa ser instalado antes de:	Python
Dependências opcionais:	Nenhum

Libpipeline

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Pkgconf
Precisa ser instalado antes de:	Man-DB
Dependências opcionais:	Nenhum

Libtool

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Autoconf, Automake, Bash, Binutils, Coreutils, File, GCC, Glibc, Grep, Make e Sed
A suíte de teste depende de:	Autoconf, Automake e Findutils
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Libxcrypt

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Perl e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Perl, Python, Shadow e Systemd
Dependências opcionais:	Nenhum

Linux

A Instalação depende de:	Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Libelf, Make, Ncurses, OpenSSL, Perl e Sed
Exigido em tempo de execução:	Nenhum
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>cpio</i> , <i>LLVM</i> (com Clang) e <i>Rust-bindgen</i>

Cabeçalhos da API do Linux

A Instalação depende de:	Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl e Sed
Exigido em tempo de execução:	Nenhum
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Lz4

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc e Make
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Python
Precisa ser instalado antes de:	Zstd e Systemd
Dependências opcionais:	Nenhum

M4

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Bash e Glibc
A suíte de teste depende de:	Diffutils
Precisa ser instalado antes de:	Autoconf e Bison
Dependências opcionais:	<i>libsigsegv</i>

Make

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Perl e Procps-ng
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Guile</i>

Man-DB

A Instalação depende de:	Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Pkgconf, Sed, Systemd e Xz
Exigido em tempo de execução:	Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline e Zlib
A suíte de teste depende de:	Util-linux
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>libseccomp</i> e <i>po4a</i>

Man-Pages

A Instalação depende de:	Bash, Coreutils, Make e Sed
Exigido em tempo de execução:	Nenhum
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

MarkupSafe

A Instalação depende de:	Python, Setuptools e Wheel
Exigido em tempo de execução:	Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Jinja2
Dependências opcionais:	Nenhum

Meson

A Instalação depende de:	Ninja, Python, Setuptools e Wheel
Exigido em tempo de execução:	Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Systemd
Dependências opcionais:	Nenhum

Mpdecimal

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make e Sed
Exigido em tempo de execução:	GCC e Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Python
Dependências opcionais:	Nenhum

MPC

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed e Texinfo
Exigido em tempo de execução:	Glibc, GMP e MPFR
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	GCC
Dependências opcionais:	Nenhum

MPFR

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed e Texinfo
Exigido em tempo de execução:	Glibc e GMP
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Gawk e GCC
Dependências opcionais:	Nenhum

Ncurses

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux e Vim
Dependências opcionais:	Nenhum

Ninja

A Instalação depende de:	Binutils, Coreutils, GCC e Python
Exigido em tempo de execução:	GCC e Glibc
A suíte de teste depende de:	<i>cmake</i>
Precisa ser instalado antes de:	Meson
Dependências opcionais:	<i>Asciidoc, Doxygen, Emacs e re2c</i>

OpenSSL

A Instalação depende de:	Binutils, Coreutils, GCC, Make e Perl
Exigido em tempo de execução:	Glibc e Perl
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Coreutils, Kmod, Linux e Systemd
Dependências opcionais:	Nenhum

Packaging

A Instalação depende de:	Flit-core e Python
Exigido em tempo de execução:	Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Wheel
Dependências opcionais:	<i>pytest</i>

Patch

A Instalação depende de:	Attr, Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make e Sed
Exigido em tempo de execução:	Attr e Glibc
A suíte de teste depende de:	Diffutils
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Ed</i>

Pcre2

A Instalação depende de:	Bash, Binutils, Bzip2, Coreutils, GCC, Glibc, GZip, Make e Readline
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Grep
Precisa ser instalado antes de:	Grep e Less
Dependências opcionais:	<i>Valgrind e libedit</i>

Perl

A Instalação depende de:	Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Libxcrypt, Make, Sed e Zlib
Exigido em tempo de execução:	GDBM, Glibc e Libxcrypt
A suíte de teste depende de:	Iana-Etc, Less e Procps-ng
Precisa ser instalado antes de:	Autoconf
Dependências opcionais:	<i>Berkeley DB</i>

Pkgconf

A Instalação depende de:	Binutils, GCC, Glibc, Meson, Ninja e Python
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Binutils, D-Bus, E2fsprogs, IProute2, Kmod, Man-DB, Procps-ng, Python, Systemd e Util-linux
Dependências opcionais:	Nenhum

Procps-ng

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Make, Ncurses Pkgconf e Systemd
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	DejaGNU
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Psmisc

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses e Sed
Exigido em tempo de execução:	Glibc e Ncurses
A suíte de teste depende de:	Expect
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Python

A Instalação depende de:	Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Libxcrypt, Make, Mpdecimal, Ncurses, OpenSSL, Pkgconf, Sed, Util-linux e Zlib
Exigido em tempo de execução:	Bzip2, Expat, Gdbm, Glibc, Libffi, Libxcrypt, Mpdecimal, Ncurses, OpenSSL e Zlib
A suíte de teste depende de:	GDB e Valgrind
Precisa ser instalado antes de:	Ninja
Dependências opcionais:	<i>Berkeley DB, libnsl e Tk</i>

Readline

A Instalação depende de:	Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed e Texinfo
Exigido em tempo de execução:	Glibc e Ncurses
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Bash, Bc e Gawk
Dependências opcionais:	Nenhum

Sed

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Acl, Attr e Glibc
A suíte de teste depende de:	Diffutils e Gawk
Precisa ser instalado antes de:	E2fsprogs, File, Libtool e Shadow
Dependências opcionais:	Nenhum

Setuptools

A Instalação depende de:	Python e Wheel
Exigido em tempo de execução:	Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Jinja2, MarkupSafe e Meson
Dependências opcionais:	Nenhum

Shadow

A Instalação depende de:	Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Libxcrypt, Make e Sed
Exigido em tempo de execução:	Glibc e Libxcrypt
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Coreutils
Dependências opcionais:	<i>CrackLib</i> e <i>Linux-PAM</i>

Sqlite

A Instalação depende de:	Bash, Binutils, GCC, Glibc, Gzip, Make, Ncurses, Readline e Zlib
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Python
Dependências opcionais:	<i>libarchive</i> e <i>libedit</i>

Systemd

A Instalação depende de:	Acl, Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libxcrypt, Lz4, Meson, OpenSSL, Pcre2, Pkgconf, Sed, Util-linux e Zstd
Exigido em tempo de execução:	Acl, Glibc, Libxcrypt, OpenSSL, Util-linux, Xz, Zlib e Zstd
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	D-Bus, E2fsprogs, Man-DB, Procps-ng e Util-linux
Dependências opcionais:	<i>AppArmor</i> , <i>audit-userspace</i> , <i>bash-completion</i> , <i>btrfs-progs</i> , <i>cURL</i> , <i>cryptsetup</i> , <i>docbook-xml</i> , <i>docbook-xsl-nons</i> , <i>Git</i> , <i>GnuTLS</i> , <i>iptables</i> , <i>jekyll</i> , <i>kexec-tools</i> , <i>libbpf</i> , <i>libdw</i> , <i>libfido2</i> , <i>libgcrypt</i> , <i>libidn2</i> , <i>libmicrohttpd</i> , <i>libpwquality</i> , <i>libseccomp</i> , <i>libxkbcommon</i> , <i>libxslt</i> , <i>Linux-PAM</i> , <i>lxml</i> , <i>make-ca</i> , <i>p11-kit</i> , <i>pefile</i> , <i>Polkit</i> , <i>pyelftools</i> , <i>qemu</i> , <i>qrencode</i> , <i>quota-tools</i> , <i>rpm</i> , <i>rsync</i> , <i>SELinux</i> , <i>Sphinx</i> , <i>systemtap</i> , <i>tpm2-tss</i> , <i>Valgrind</i> , <i>Xen</i> e <i>zsh</i>

Tar

A Instalação depende de:	Acl, Attr, Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed e Texinfo
Exigido em tempo de execução:	Acl, Attr, Bzip2, Glibc, Gzip e Xz
A suíte de teste depende de:	Autoconf, Diffutils, Findutils, Gawk e Gzip
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Tcl

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed
Exigido em tempo de execução:	Glibc e Zlib
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

Texinfo

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch e Sed
Exigido em tempo de execução:	Glibc e Ncurses
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	SWIG

Util-linux

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Pkgconf, Sed, Systemd e Zlib
Exigido em tempo de execução:	Glibc, Ncurses, Readline, Systemd e Zlib
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Asciidoctor, Libcap-NG, libeconf, libuser, libutempter, Linux-PAM, smartmontools, po4a e slang</i>

Vim

A Instalação depende de:	Acl, Attr, Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed
Exigido em tempo de execução:	Acl, Attr, Glibc, Python, Ncurses e Tcl
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>Xorg, GTK+2, LessTif, Ruby e GPM</i>

Wheel

A Instalação depende de:	Python, Flit-core e packaging
Exigido em tempo de execução:	Python
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Jinja2, MarkupSafe, Meson e Setuptools
Dependências opcionais:	Nenhum
Exigido em tempo de execução:	Expat, Glibc e Perl
A suíte de teste depende de:	Perl
Precisa ser instalado antes de:	Intltool
Dependências opcionais:	<i>LWP::UserAgent</i>

Xz

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc e Make
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	File, GRUB, Kmod, Libelf, Man-DB e Systemd
Dependências opcionais:	Nenhum

Zlib

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Make e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	File, Kmod, Libelf, Perl, Python e Util-linux
Dependências opcionais:	Nenhum

Zstd

A Instalação depende de:	Binutils, Coreutils, GCC, Glibc, Gzip, Lz4, Make, Xz e Zlib
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Binutils, File, GCC, Kmod, Libelf, Python e Systemd
Dependências opcionais:	Nenhum

Apêndice D. Licenças do LFS

Este livro está licenciado sob a Licença Creative Commons Atribuição-Usa Não Comercial-Compartilhamento pela mesma licença 2.0.

As instruções de computador podem ser extraídas a partir do livro sob a Licença do MIT.

D.1. Licença da Creative Commons

Código Jurídico da Creative Commons

Atribuição - Usa não-Comercial - Compartilhamento pela mesma licença 2.0



Importante

A INSTITUIÇÃO CREATIVE COMMONS NÃO É UM ESCRITÓRIO DE ADVOCACIA E NÃO PRESTA SERVIÇOS JURÍDICOS. A DISTRIBUIÇÃO DESTA LICENÇA NÃO ESTABELECE QUALQUER RELAÇÃO ADVOCATÍCIA. A CREATIVE COMMONS DISPONIBILIZA ESTA INFORMAÇÃO "NO ESTADO EM QUE SE ENCONTRA". A CREATIVE COMMONS NÃO FAZ QUALQUER GARANTIA QUANTO ÀS INFORMAÇÕES DISPONIBILIZADAS E SE EXONERA DE QUALQUER RESPONSABILIDADE POR DANOS RESULTANTES DO SEU USO.

Licença

A OBRA (CONFORME DEFINIDA ABAIXO) É DISPONIBILIZADA DE ACORDO COM OS TERMOS DESTA LICENÇA PÚBLICA CREATIVE COMMONS ("CCPL" OU "LICENÇA"). A OBRA É PROTEGIDA POR DIREITO AUTURAL E (OU) OUTRAS LEIS APLICÁVEIS. QUALQUER USO DA OBRA QUE NÃO O AUTORIZADO SOB ESTA LICENÇA OU PELA LEGISLAÇÃO AUTURAL É PROIBIDO.

AO EXERCER QUAISQUER DOS DIREITOS À OBRA AQUI CONCEDIDOS, VOCÊ ACEITA E CONCORDA FICAR OBRIGADO(A) NOS TERMOS DESTA LICENÇA. O LICENCIANTE CONCEDE A VOCÊ OS DIREITOS AQUI CONTIDOS EM CONTRAPARTIDA À SUA ACEITAÇÃO DESTES TERMOS E CONDIÇÕES.

1. Definições

- a. "Obra Coletiva" significa uma obra, tal como uma edição periódica, antologia ou enciclopédia, na qual a Obra em sua totalidade e de forma inalterada, em conjunto com um número de outras contribuições, constituindo obras independentes e separadas em si mesmas, são agregadas em um trabalho coletivo. Uma obra que constitua uma Obra Coletiva não será considerada Obra Derivada (conforme definido abaixo) para os propósitos desta licença.
- b. "Obra Derivada" significa uma obra baseada sobre a Obra ou sobre a Obra e outras obras pré existentes, tal como uma tradução, arranjo musical, dramatização, romantização, versão de filme, gravação de som, reprodução de obra artística, resumo, condensação ou qualquer outra forma na qual a Obra possa ser refeita, transformada ou adaptada, com a exceção de que uma obra que constitua uma Obra Coletiva não será considerada Obra Derivada para fins desta licença. Para evitar dúvidas, quando a Obra for uma composição musical ou gravação de som, a sincronização da Obra em relação cronometrada com uma imagem em movimento ("synching") será considerada uma Obra Derivada para os propósitos desta licença.
- c. "Licenciante" significa a pessoa física ou a jurídica que oferece a Obra sob os termos desta Licença.
- d. "Autor(a) Original" significa a pessoa física ou jurídica que criou a Obra.
- e. "Obra" significa a obra autoral, passível de proteção pelo direito autoral, oferecida sob os termos desta Licença.

- f. "Você" significa a pessoa física ou jurídica exercendo direitos sob esta Licença que não tenha previamente violado os termos desta Licença com relação à Obra, ou que tenha recebido permissão expressa do(a) Licenciante para exercer direitos sob esta Licença apesar de uma violação prévia.
 - g. "Elementos da Licença" significa os principais atributos da licença correspondente, conforme escolhidos pelo(a) Licenciante e indicados no título desta Licença: Atribuição, Não-comercial, Compartilhamento pela Mesma Licença.
2. Direitos de Uso Legítimo. Nada nesta licença é destinado a reduzir, limitar ou restringir quaisquer direitos emergentes do uso legítimo, primeira venda ou outras limitações sobre os direitos exclusivos do titular de direitos autorais sob a legislação autoral ou quaisquer outras leis aplicáveis.
 3. Concessão da Licença. Sujeita aos termos e condições desta Licença, o(a) Licenciante concede a Você uma licença de abrangência mundial, sem royalties, não-exclusiva, perpétua (pela duração do direito autoral aplicável), para exercer os direitos sobre a Obra definidos abaixo:
 - a. reproduzir a Obra, incorporar a Obra em uma ou mais Obras Coletivas e reproduzir a Obra quando incorporada em Obra Coletiva;
 - b. para criar e reproduzir Obras Derivadas;
 - c. para distribuir cópias ou gravações da Obra, exibir publicamente, executar publicamente e executar publicamente por meio de uma transmissão de áudio digital a Obra, inclusive quando incorporada em Obras Coletivas;
 - d. para distribuir cópias ou gravações de Obras Derivadas, exibir publicamente, executar publicamente e executar publicamente por meio de uma transmissão digital de áudio Obras Derivadas;

Os direitos acima podem ser exercidos em todas as mídias e formatos, independente de serem conhecidos agora ou concebidos posteriormente. Os direitos acima incluem o direito de fazer modificações que forem tecnicamente necessárias para exercer os direitos em outras mídias, meios e formatos. Todos os direitos não concedidos expressamente pelo(a) Licenciante ficam aqui reservados, incluindo, mas não se limitando, os direitos definidos nas Seções 4(e) e 4(f).

4. Restrições. A licença concedida na Seção 3 acima está expressamente sujeita e limitada aos seguintes termos:
 - a. Você pode distribuir, exibir publicamente, executar publicamente ou executar publicamente digitalmente a Obra somente sob os termos desta Licença, e Você precisa incluir uma cópia de, ou o Uniform Resource Identifier para, esta Licença com cada cópia ou fonograma da Obra que Você distribuir, exibir publicamente, executar publicamente ou executar publicamente digitalmente. Você não pode oferecer ou impor quaisquer termos sobre a Obra que alterem ou restrinjam os termos desta Licença ou o exercício dos(as) destinatários(as) dos direitos aqui concedidos. Você não pode sublicenciar a Obra. Você precisa manter intactos todos os avisos que se referem a esta Licença e à isenção de garantias. Você não pode distribuir, exibir publicamente, executar publicamente ou executar publicamente digitalmente a Obra com quaisquer medidas tecnológicas que controlem o acesso ou uso da Obra de uma maneira inconsistente com os termos deste Contrato de Licença. O acima se aplica para a Obra conforme incorporada em uma Obra Coletiva, mas isso não exige que a Obra Coletiva, além da própria Obra, seja submetida aos termos desta Licença. Se Você criar uma Obra Coletiva, mediante notificação originária de qualquer Licenciante, Você precisará, na medida do possível, remover da Obra Coletiva qualquer referência para tal Licenciante ou para o(a) Autor(a) Original, conforme solicitado. Se Você criar uma Obra Derivada, mediante notificação originária de qualquer Licenciante, Você precisará, na medida do possível, remover da Obra Derivada qualquer referência para tal Licenciante ou para o(a) Autor(a) Original, conforme solicitado.
 - b. Você pode distribuir, exibir publicamente, executar publicamente ou executar publicamente por meios digitais uma Obra Derivada somente sob os termos desta Licença, ou de uma versão posterior desta licença com os mesmos Elementos da Licença desta licença, ou de uma licença do internacional da Creative Commons (iCommons) que contenha os mesmos Elementos da Licença desta Licença (por exemplo, Atribuição, Uso Não Comercial, Compartilhamento pela Mesma Licença Japão). Você deve incluir uma

cópia desta licença ou de outra licença especificada na sentença anterior, ou o Identificador Uniformizado de Recursos (Uniform Resource Identifier) para esta licença ou de outra licença especificada na sentença anterior, com cada cópia ou gravação de cada Obra Derivada que Você distribuir, exibir publicamente, executar publicamente ou executar publicamente por meios digitais. Você não poderá oferecer ou impor quaisquer termos sobre a Obra Derivada que alterem ou restrinjam os termos desta Licença ou o exercício dos direitos aqui concedidos para os(as) destinatários(as), e Você deverá manter intactas todas as informações que se refiram a esta Licença e à exclusão de garantias. Você não poderá distribuir, exibir publicamente, executar publicamente ou executar publicamente por meios digitais a Obra Derivada com qualquer medida tecnológica que controle o acesso ou o uso da Obra de maneira inconsistente com os termos deste Acordo de Licença. O disposto acima se aplica à Obra Derivada quando incorporada em uma Obra Coletiva, mas isso não requer que a Obra Coletiva, à parte da Obra em si, esteja sujeita aos termos desta Licença.

- c. Você não poderá exercer nenhum dos direitos acima concedidos a Você na Seção 3 de qualquer maneira que seja predominantemente intencionada ou direcionada à obtenção de vantagem comercial ou compensação monetária privada. A troca da Obra por outros materiais protegidos por direito autoral por intermédio de compartilhamento digital de arquivos ou de outras formas não deverá ser considerada como intencionada ou direcionada à obtenção de vantagens comerciais ou compensação monetária privada, desde que não haja pagamento de nenhuma compensação monetária com relação à troca de obras protegidas por direito de autor.
 - d. Se Você distribuir, exibir publicamente, executar publicamente ou executar publicamente por meios digitais a Obra ou qualquer Obra Derivada ou Obra Coletiva, Você deve manter intactas todas as informações relativas a direitos autorais sobre a Obra e atribuir para o(a) Autor(a) Original crédito razoável com relação ao meio ou mídia que Você está utilizando, por intermédio da veiculação do nome (ou pseudônimo, se for o caso) do(a) Autor(a) Original, se fornecido; o título da Obra, se fornecido; na medida do razoável, o Identificador Uniformizado de Recursos (URI) que o(a) Licenciante especificar para estar associado à Obra, se houver, exceto se o URI não se referir ao aviso de direitos autorais ou à informação sobre o regime de licenciamento da Obra; e no caso de Obra Derivada, crédito identificando o uso da Obra na Obra Derivada (exemplo: "Tradução Francesa da Obra de Autor(a) Original", ou "Roteiro baseado na Obra original de Autor(a) Original"). Tal crédito pode ser implementado de qualquer forma razoável; entretanto, no caso de Obra Derivada ou Obra Coletiva, esse crédito aparecerá no mínimo onde qualquer outro crédito comparável de autoria aparece e de modo ao menos tão proeminente quanto esse outro crédito de autoria comparável.
 - e. Para evitar dúvidas, quando a Obra for uma composição musical:
 - i. Royalties de Desempenho Sob Licenças Gerais. O(A) Licenciante reserva-se o direito exclusivo de cobrar, seja individualmente ou por meio de uma sociedade de direitos de execução (por exemplo, ASCAP, BMI, SESAC), royalties pela execução pública ou pela execução digital pública (por exemplo, webcast) da Obra, se essa execução for destinada principalmente ou dirigida em direção a vantagens comerciais ou compensação monetária privada.
 - ii. Direitos Mecânicos e Royalties Estatutários. O(A) Licenciante reserva-se o direito exclusivo de cobrar, seja individualmente ou por meio de uma agência de direitos musicais ou agente designado (por exemplo, Agência Harry Fox), royalties por qualquer gravação fonográfica que Você criar a partir da Obra (versão cover) e distribuir, sujeita à licença compulsória criada pela 17 USC Seção 115 da Lei de Direitos Autorais dos Estados Unidos da América do Norte (ou equivalente em outras jurisdições), se a distribuição de tal versão cover for principalmente destinada ou direcionada a vantagens comerciais ou compensação monetária privada.
6. Direitos de Webcasting e Royalties Estatutários. Para evitar dúvidas, quando a Obra for uma gravação de som, o(a) Licenciante reserva-se o direito exclusivo de cobrar, seja individualmente ou por intermédio de uma sociedade de direitos de execução (por exemplo, SoundExchange), royalties pela execução digital pública (por exemplo, webcast) da Obra, sujeito à licença compulsória criada pela 17 USC Seção 114 da Lei de Direitos Autorais dos Estados Unidos da América do Norte (ou equivalente em outras jurisdições), se a Tua execução digital pública for principalmente destinada ou direcionada a vantagens comerciais ou compensação monetária privada.

- f. Direitos de Webcast e Royalties Estatutários. Para evitar dúvidas, quando a Obra for uma gravação de som, o(a) Licenciante reserva-se o direito exclusivo de coletar, seja individualmente ou por meio de uma sociedade de direitos de execução (por exemplo, SoundExchange), royalties pela execução digital pública (por exemplo, webcast) da Obra, sujeita à licença compulsória criada pela 17 USC Seção 114 da Lei de Direitos Autorais dos Estados Unidos da América do Norte (ou equivalente em outras jurisdições), se a Tua execução digital pública for principalmente destinada ou direcionada para vantagem comercial ou compensação monetária privada.

5. Declarações, Garantias e Isenção de Responsabilidade

EXCETO QUANDO FOR DE OUTRA FORMA MUTUAMENTE ACORDADO PELAS PARTES POR ESCRITO, O(A) LICENCIANTE OFERECE A OBRA NO ESTADO EM QUE SE ENCONTRA (AS IS) E NÃO PRESTA QUAISQUER GARANTIAS OU DECLARAÇÕES DE QUALQUER ESPÉCIE RELATIVAS À OBRA, SEJAM ELAS EXPRESSAS OU IMPLÍCITAS, DECORRENTES DA LEI OU QUAISQUER OUTRAS, INCLUINDO, SEM LIMITAÇÃO, QUAISQUER GARANTIAS SOBRE A TITULARIDADE DA OBRA, ADEQUAÇÃO PARA QUAISQUER PROPÓSITOS, NÃO-VIOLAÇÃO DE DIREITOS, OU INEXISTÊNCIA DE QUAISQUER DEFEITOS LATENTES, ACURACIDADE, PRESENÇA OU AUSÊNCIA DE ERROS, SEJAM ELES APARENTES OU OCULTOS. EM JURISDIÇÕES QUE NÃO ACEITEM A EXCLUSÃO DE GARANTIAS IMPLÍCITAS, ESSAS EXCLUSÕES PODEM NÃO SE APLICAR A VOCÊ.

6. Limitação de Responsabilidade. EXCETO NA EXTENSÃO EXIGIDA PELA LEI APLICÁVEL, EM NENHUMA CIRCUNSTÂNCIA O(A) LICENCIANTE SERÁ RESPONSÁVEL PARA COM VOCÊ POR QUAISQUER DANOS, ESPECIAIS, INCIDENTAIS, CONSEQUENCIAIS, PUNITIVOS OU EXEMPLARES, ORIUNDOS DESTA LICENÇA OU DO USO DA OBRA, MESMO QUE O(A) LICENCIANTE TENHA SIDO AVISADO(A) SOBRE A POSSIBILIDADE DE TAIS DANOS.

7. Terminação

- a. Esta Licença e os direitos aqui concedidos terminarão automaticamente no caso de qualquer violação dos termos desta Licença por Você. Pessoas físicas ou jurídicas que tenham recebido Obras Derivadas ou Obras Coletivas de Você sob esta Licença, entretanto, não terão suas licenças terminadas desde que tais pessoas físicas ou jurídicas permaneçam em total cumprimento com essas licenças. As Seções 1, 2, 5, 6, 7 e 8 subsistirão a qualquer terminação desta Licença.
- b. Sujeito aos termos e condições dispostos acima, a licença aqui concedida é perpétua (pela duração do direito autoral aplicável à Obra). Não obstante o disposto acima, o(a) Licenciante reserva-se o direito de difundir a Obra sob termos diferentes de licença ou de cessar a distribuição da Obra a qualquer momento; desde que, no entanto, quaisquer destas ações não sirvam como meio de retratação desta Licença (ou de qualquer outra licença que tenha sido concedida sob os termos desta Licença, ou que deva ser concedida sob os termos desta Licença) e esta Licença continuará válida e eficaz a não ser que seja terminada de acordo com o disposto acima.

8. Outras Disposições

- a. Cada vez que Você distribuir ou executar publicamente por meios digitais a Obra ou uma Obra Coletiva, o(a) Licenciante oferece ao destinatário uma licença da Obra nos mesmos termos e condições que a licença concedida a Você sob esta Licença.
- b. Cada vez que Você distribuir ou executar publicamente por meios digitais uma Obra Derivada, o(a) Licenciante oferece ao destinatário uma licença à Obra original nos mesmos termos e condições que foram concedidos a Você sob esta Licença.
- c. Se qualquer disposição desta Licença for tida como inválida ou não-executável sob a lei aplicável, isso não afetará a validade ou a possibilidade de execução do restante dos termos desta Licença e, sem a necessidade de qualquer ação adicional das partes deste acordo, tal disposição será reformada na mínima extensão necessária para tal disposição tornar-se válida e executável.

- d. Nenhum termo ou disposição desta Licença será considerado renunciado e nenhuma violação será considerada consentida, a não ser que tal renúncia ou consentimento seja feita por escrito e assinada pela parte que será afetada por tal renúncia ou consentimento.
- e. Esta Licença representa o acordo integral entre as partes com respeito à Obra aqui licenciada. Não há entendimentos, acordos ou declarações relativas à Obra que não estejam especificadas aqui. O(A) Licenciante não será obrigado(a) por nenhuma disposição adicional que possa aparecer em quaisquer comunicações provenientes de Você. Esta Licença não pode ser modificada sem o mútuo acordo, por escrito, entre o(a) Licenciante e Você.



Importante

A Creative Commons não é uma parte desta Licença e não presta qualquer garantia relacionada à Obra. A Creative Commons não será responsável perante Você ou qualquer outra parte por quaisquer danos, incluindo, sem limitação, danos gerais, especiais, incidentais ou consequentes, originados com relação a esta licença. Não obstante as duas frases anteriores, se a Creative Commons tiver expressamente se identificado como a Licenciante, ela deverá ter todos os direitos e obrigações de Licenciante.

Exceto para o propósito delimitado de indicar ao público que a Obra é licenciada sob a CCPL (Licença Pública Creative Commons), nenhuma parte deverá utilizar a marca Creative Commons ou qualquer outra marca ou logo relacionado à Creative Commons sem consentimento prévio e por escrito da Creative Commons. Qualquer uso permitido deverá ser de acordo com as diretrizes da Creative Commons de utilização da marca então válidas, conforme sejam publicadas no sítio da web dela ou de outro modo disponibilizadas periodicamente mediante solicitação.

A Creative Commons pode ser contactada em <http://creativecommons.org/>.

D.2. A Licença do MIT

Direitos autorais © 1999-2026 Gerard Beekmans

Permissão é aqui concedida, gratuitamente, para qualquer pessoa que obtenha uma cópia deste software e arquivos de documentação associados (o "Software"), para lidar com o Software sem restrição, incluindo, sem limitação, os direitos para usar, copiar, modificar, mesclar, publicar, distribuir, sublicenciar, e (ou) vender cópias do Software, e para permitir para as pessoas para quem o Software for fornecido para fazer o mesmo, sujeito às seguintes condições:

O aviso de direitos autorais acima e este aviso de permissão deveria ser incluído em todas as cópias ou porções substanciais do Software.

O SOFTWARE É FORNECIDO “NO ESTADO EM QUE SE ENCONTRA”, SEM GARANTIAS DE QUALQUER ESPÉCIE, EXPLÍCITAS OU IMPLÍCITAS, INCLUINDO, PORÉM NÃO LIMITADA A, AS GARANTIAS DE COMERCIALIZAÇÃO, ADEQUAÇÃO PARA UM PROPÓSITO PARTICULAR E NÃO-VIOLAÇÃO. EM NENHUMA CIRCUNSTÂNCIA OS AUTORES OU TITULARES DE DIREITOS AUTORAIS SERÃO RESPONSÁVEIS POR QUAISQUER ALEGAÇÕES, DANOS OU OUTRA RESPONSABILIDADE, SEJA EM UMA AÇÃO DE CONTRATO, ATO ILÍCITO OU DE OUTRA FORMA, DECORRENTE DE, OU EM CONEXÃO COM, O SOFTWARE OU O USO OU OUTRAS NEGOCIAÇÕES NO SOFTWARE.

Índice Remissivo

- Acl: 147
- Attr: 146
- Autoconf: 183
- Automake: 184
- Bash: 171
 - ferramentas: 68
- Bash: 171
 - ferramentas: 68
- Bc: 130
- Binutils: 139
 - ferramentas, passagem 1: 52
 - ferramentas, passagem 2: 81
- Binutils: 139
 - ferramentas, passagem 1: 52
 - ferramentas, passagem 2: 81
- Binutils: 139
 - ferramentas, passagem 1: 52
 - ferramentas, passagem 2: 81
- Bison: 169
 - ferramentas: 91
- Bison: 169
 - ferramentas: 91
- Bzip2: 118
- Coreutils: 201
 - ferramentas: 69
- Coreutils: 201
 - ferramentas: 69
- D-Bus: 238
- DejaGNU: 136
- Diffutils: 206
 - ferramentas: 70
- Diffutils: 206
 - ferramentas: 70
- E2fsprogs: 250
- Expat: 176
- Expect: 134
- File: 124
 - ferramentas: 71
- File: 124
 - ferramentas: 71
- Findutils: 207
 - ferramentas: 72
- Findutils: 207
 - ferramentas: 72
- Flex: 131
- Flit-core: 195
- Gawk: 155
 - ferramentas: 73
- Gawk: 155
 - ferramentas: 73
- GCC: 156
 - ferramentas, libstdc++ passagem 1: 62
 - ferramentas, passagem 1: 54
 - ferramentas, passagem 2: 82
- GCC: 156
 - ferramentas, libstdc++ passagem 1: 62
 - ferramentas, passagem 1: 54
 - ferramentas, passagem 2: 82
- GCC: 156
 - ferramentas, libstdc++ passagem 1: 62
 - ferramentas, passagem 1: 54
 - ferramentas, passagem 2: 82
- GCC: 156
 - ferramentas, libstdc++ passagem 1: 62
 - ferramentas, passagem 1: 54
 - ferramentas, passagem 2: 82
- GDBM: 174
- Gettext: 167
 - ferramentas: 90
- Gettext: 167
 - ferramentas: 90
- Glibc: 109
 - ferramentas: 58
- Glibc: 109
 - ferramentas: 58
- GMP: 142
- Gperf: 175
- Grep: 170
 - ferramentas: 74
- Grep: 170
 - ferramentas: 74
- Groff: 208
- GRUB: 211
- Gzip: 215
 - ferramentas: 75
- Gzip: 215
 - ferramentas: 75
- Iana-Etc: 108
- Inetutils: 177
- IPRoute2: 216
- Jinja2: 231
- Kbd: 218
- Kmod: 200
- Less: 179
- Libcap: 148
- Libelf: 187
- libffi: 188

Libpipeline: 221
 Libtool: 173
 Libxcrypt: 149
 Linux: 277
 ferramentas, cabeçalhos da API: 57
 Linux: 277
 ferramentas, cabeçalhos da API: 57
 Lz4: 122
 M4: 129
 ferramentas: 65
 M4: 129
 ferramentas: 65
 Make: 222
 ferramentas: 76
 Make: 222
 ferramentas: 76
 Man-DB: 240
 Páginas-Manual: 107
 MarkupSafe: 230
 Meson: 199
 MPC: 145
 mpdecimal: 94, 191
 mpdecimal: 94, 191
 MPFR: 144
 Ncurses: 162
 ferramentas: 66
 Ncurses: 162
 ferramentas: 66
 Ninja: 137
 OpenSSL: 185
 packaging: 196
 Patch: 223
 ferramentas: 77
 Patch: 223
 ferramentas: 77
 Pcre2: 127
 Perl: 180
 ferramentas: 92
 Perl: 180
 ferramentas: 92
 Pkgconf: 138
 Procps-ng: 243
 Psmisc: 166
 Python: 192
 temporário: 95
 Python: 192
 temporário: 95
 Readline: 125
 Sed: 165
 ferramentas: 78
 Sed: 165
 ferramentas: 78
 Setuptools: 198
 Shadow: 151
 configurando: 152
 Shadow: 151
 configurando: 152
 Sqlite: 189
 systemd: 232
 Tar: 224
 ferramentas: 79
 Tar: 224
 ferramentas: 79
 Tcl: 132
 Texinfo: 225
 temporário: 96
 Texinfo: 225
 temporário: 96
 Udev
 uso: 260
 Util-linux: 245
 ferramentas: 97
 Util-linux: 245
 ferramentas: 97
 Vim: 227
 wheel: 197
 Xz: 120
 ferramentas: 80
 Xz: 120
 ferramentas: 80
 Zlib: 93, 117
 Zlib: 93, 117
 zstd: 123

 [: 201, 202
 accessdb: 240, 241
 aclocal: 184, 184
 aclocal-1.18: 184, 184
 addftinfo: 208, 208
 addpart: 245, 246
 addr2line: 139, 140
 afmtodit: 208, 208
 agetty: 245, 246
 apropos: 240, 241
 ar: 139, 140
 as: 139, 140
 attr: 146, 146
 autoconf: 183, 183
 autoheader: 183, 183
 autom4te: 183, 183

automake: 184, 184
 automake-1.18: 184, 184
 autopoint: 167, 167
 autoreconf: 183, 183
 autoscan: 183, 183
 autoupdate: 183, 183
 awk: 155, 155
 b2sum: 201, 202
 badblocks: 250, 251
 base64: 201, 202, 201, 202
 base64: 201, 202, 201, 202
 basename: 201, 202
 basenc: 201, 202
 bash: 171, 172
 bashbug: 171, 172
 bc: 130, 130
 bison: 169, 169
 blkdiscard: 245, 246
 blkid: 245, 246
 blkzone: 245, 246
 blockdev: 245, 246
 bomtool: 138, 138
 bootctl: 232, 234
 bridge: 216, 216
 bunzip2: 118, 119
 busctl: 232, 234
 bzipcat: 118, 119
 bzcmp: 118, 119
 bzdiff: 118, 119
 bzegrep: 118, 119
 bzfgrep: 118, 119
 bzgrep: 118, 119
 bzip2: 118, 119
 bzip2recover: 118, 119
 bzless: 118, 119
 bzipmore: 118, 119
 c++: 156, 160
 c++filt: 139, 140
 cal: 245, 246
 capsh: 148, 148
 captoinfo: 162, 163
 cat: 201, 202
 catman: 240, 241
 cc: 156, 160
 cfdisk: 245, 246
 chacl: 147, 147
 chage: 151, 153
 chattr: 250, 251
 chcpu: 245, 246
 chem: 208, 208
 chfn: 151, 153
 chgpasswd: 151, 153
 chgrp: 201, 202
 chmem: 245, 246
 chmod: 201, 202
 choom: 245, 246
 chown: 201, 202
 chpasswd: 151, 153
 chroot: 201, 202
 chrt: 245, 246
 chsh: 151, 153
 chvt: 218, 219
 cksum: 201, 203
 clear: 162, 163
 cmp: 206, 206
 col: 245, 246
 colcrt: 245, 246
 colrm: 245, 246
 column: 245, 246
 comm: 201, 203
 compile_et: 250, 251
 coredumpctl: 232, 234
 corelist: 180, 181
 cp: 201, 203
 cpan: 180, 181
 cpp: 156, 160
 csplit: 201, 203
 ctrlaltdel: 245, 246
 ctstat: 216, 216
 cut: 201, 203
 c_rehash: 185, 186
 date: 201, 203
 dbus-cleanup-sockets: 238, 238
 dbus-daemon: 238, 238
 dbus-launch: 238, 239
 dbus-monitor: 238, 239
 dbus-run-session: 238, 239
 dbus-send: 238, 239
 dbus-test-tool: 238, 239
 dbus-update-activation-environment: 238, 239
 dbus-uuidgen: 238, 239
 dc: 130, 130
 dd: 201, 203
 deallocvt: 218, 219
 debugfs: 250, 251
 dejagnu: 136, 136
 delpart: 245, 246
 depmod: 200, 200
 df: 201, 203
 diff: 206, 206

diff3: 206, 206
dir: 201, 203
dircolors: 201, 203
dirname: 201, 203
dmesg: 245, 246
dnsdomainname: 177, 178
du: 201, 203
dumpe2fs: 250, 251
dumpkeys: 218, 219
e2freefrag: 250, 251
e2fsck: 250, 251
e2image: 250, 251
e2label: 250, 251
e2mmpstatus: 250, 251
e2scrub: 250, 251
e2scrub_all: 250, 251
e2undo: 250, 251
e4crypt: 250, 251
e4defrag: 250, 251
echo: 201, 203
egrep: 170, 170
eject: 245, 246
elfedit: 139, 140
enc2xs: 180, 181
encguess: 180, 181
env: 201, 203
envsubst: 167, 167
eqn: 208, 208
eqn2graph: 208, 208
ex: 227, 229
expand: 201, 203
expect: 134, 135
expr: 201, 203
factor: 201, 203
faillog: 151, 153
fallocate: 245, 246
false: 201, 203
fdisk: 245, 247
fgconsole: 218, 219
fgrep: 170, 170
file: 124, 124
filefrag: 250, 251
fincore: 245, 247
find: 207, 207
findfs: 245, 247
findmnt: 245, 247
flex: 131, 131
flex++: 131, 131
flock: 245, 247
fmt: 201, 203
fold: 201, 203
free: 243, 243
fsck: 245, 247
fsck.cramfs: 245, 247
fsck.ext2: 250, 251
fsck.ext3: 250, 251
fsck.ext4: 250, 252
fsck.minix: 245, 247
fsfreeze: 245, 247
fstrim: 245, 247
ftp: 177, 178
fuser: 166, 166
g++: 156, 160
gawk: 155, 155
gawk-5.4.1: 155, 155
gcc: 156, 160
gc-ar: 156, 160
gc-nm: 156, 160
gc-ranlib: 156, 160
gcov: 156, 160
gcov-dump: 156, 160
gcov-tool: 156, 160
gdbmtool: 174, 174
gdbm_dump: 174, 174
gdbm_load: 174, 174
gdiffmk: 208, 208
gencat: 109, 115
genl: 216, 216
getcap: 148, 148
getconf: 109, 115
getent: 109, 115
getfacl: 147, 147
getfattr: 146, 146
getkeycodes: 218, 219
getopt: 245, 247
getpcaps: 148, 148
getsubids: 151, 153
gettext: 167, 167
gettext.sh: 167, 167
gettextize: 167, 167
glilypond: 208, 208
gpasswd: 151, 153
gperf: 175, 175
gperl: 208, 208
gpinyin: 208, 208
gprof: 139, 140
gprofng: 139, 140
grap2graph: 208, 209
grep: 170, 170
grn: 208, 209

grodvi: 208, 209
 groff: 208, 209
 groffer: 208, 209
 grog: 208, 209
 grolbp: 208, 209
 grolj4: 208, 209
 gropdf: 208, 209
 groups: 208, 209
 grotty: 208, 209
 groupadd: 151, 153
 groupdel: 151, 153
 groupmod: 151, 153
 groups: 201, 203
 grpck: 151, 153
 grpconv: 151, 153
 grpunconv: 151, 153
 grub-bios-setup: 211, 213
 grub-editenv: 211, 213
 grub-file: 211, 213
 grub-fstest: 211, 213
 grub-glue-efi: 211, 213
 grub-install: 211, 213
 grub-kbdcomp: 211, 213
 grub-macbless: 211, 213
 grub-menulst2cfg: 211, 213
 grub-mkconfig: 211, 213
 grub-mkimage: 211, 213
 grub-mklayout: 211, 213
 grub-mknetdir: 211, 213
 grub-mkpasswd-pbkdf2: 211, 213
 grub-mkrelpath: 211, 213
 grub-mkrescue: 211, 213
 grub-mkstandalone: 211, 213
 grub-ofpathname: 211, 213
 grub-probe: 211, 213
 grub-reboot: 211, 213
 grub-render-label: 211, 213
 grub-script-check: 211, 214
 grub-set-default: 211, 214
 grub-setup: 211, 214
 grub-syslinux2cfg: 211, 214
 gunzip: 215, 215
 gzexe: 215, 215
 gzip: 215, 215
 h2ph: 180, 181
 h2xs: 180, 181
 parar: 232, 234
 hardlink: 245, 247
 head: 201, 203
 hexdump: 245, 247
 hostid: 201, 203
 hostname: 177, 178
 hostnamectl: 232, 234
 hpftodit: 208, 209
 hwclock: 245, 247
 i386: 245, 247
 iconv: 109, 115
 iconvconfig: 109, 115
 id: 201, 203
 idle3: 192
 ifconfig: 177, 178
 ifnames: 183, 183
 ifstat: 216, 216
 indxbib: 208, 209
 info: 225, 225
 infocmp: 162, 163
 infotocap: 162, 163
 init: 232, 234
 insmod: 200, 200
 install: 201, 203
 install-info: 225, 226
 instmodsh: 180, 181
 ionice: 245, 247
 ip: 216, 216
 ipcmk: 245, 247
 ipcrm: 245, 247
 ipcs: 245, 247
 irqtop: 245, 247
 isosize: 245, 247
 join: 201, 203
 journalctl: 232, 235
 json_pp: 180, 181
 kbdinfo: 218, 219
 kbdrate: 218, 219
 kbd_mode: 218, 219
 kernel-install: 232, 235
 kill: 245, 247
 killall: 166, 166
 kmod: 200, 200
 last: 245, 247
 lastb: 245, 247
 ld: 139, 140
 ld.bfd: 139, 140
 ldattach: 245, 247
 ldconfig: 109, 115
 ldd: 109, 115
 lddlibc4: 109, 115
 less: 179, 179
 lessecho: 179, 179
 lesskey: 179, 179

lex: 131, 131	lzgrep: 120, 120
lexgrog: 240, 241	lzless: 120, 120
lfskernel-7.1.8: 277, 284	lzma: 120, 120
libasan: 156, 160	lzmadec: 120, 120
libatomic: 156, 160	lzmainfo: 120, 120
libcc1: 156, 160	lzmore: 120, 120
libnetcfg: 180, 181	m4: 129, 129
libtool: 173, 173	machinectl: 232, 235
libtoolize: 173, 173	make: 222, 222
link: 201, 203	makedb: 109, 115
linux32: 245, 247	makeinfo: 225, 226
linux64: 245, 247	man: 240, 241
lkbib: 208, 209	man-recode: 240, 241
ln: 201, 203	mandb: 240, 242
lnstat: 216, 217	manpath: 240, 242
loadkeys: 218, 219	mapscrn: 218, 219
loadunimap: 218, 219	mcookie: 245, 248
locale: 109, 115	md5sum: 201, 203
localectl: 232, 235	mesg: 245, 248
localedef: 109, 115	meson: 199, 199
locate: 207, 207	mkdir: 201, 203
logger: 245, 247	mke2fs: 250, 252
login: 151, 154	mkfifo: 201, 204
loginctl: 232, 235	mkfs: 245, 248
logname: 201, 203	mkfs.bfs: 245, 248
logsave: 250, 252	mkfs.cramfs: 245, 248
look: 245, 247	mkfs.ext2: 250, 252
lookbib: 208, 209	mkfs.ext3: 250, 252
losetup: 245, 247	mkfs.ext4: 250, 252
ls: 201, 203	mkfs.minix: 245, 248
lsattr: 250, 252	mklost+found: 250, 252
lsblk: 245, 247	mknod: 201, 204
lscpu: 245, 247	mkswap: 245, 248
lsfd: 245, 247	mktemp: 201, 204
lsipc: 245, 247	mk_cmds: 250, 252
lsirq: 245, 248	mmroff: 208, 209
lslocks: 245, 248	modinfo: 200, 200
lslogins: 245, 248	modprobe: 200, 200
lsmem: 245, 248	more: 245, 248
lsmod: 200, 200	mount: 245, 248
lsns: 245, 248	mountpoint: 245, 248
lto-dump: 156, 160	msgattrib: 167, 167
lz4: 122, 122	msgcat: 167, 167
lz4c: 122, 122	msgcmp: 167, 167
lz4cat: 122, 122	msgcomm: 167, 168
lzcac: 120, 120	msgconv: 167, 168
lzcmp: 120, 120	msgen: 167, 168
lzdiff: 120, 120	msgexec: 167, 168
lzgrep: 120, 120	msgfilter: 167, 168
lzfgrep: 120, 120	msgfmt: 167, 168

msggrep: 167, 168
 msginit: 167, 168
 msgmerge: 167, 168
 msgunfmt: 167, 168
 msguniq: 167, 168
 mtrace: 109, 115
 mv: 201, 204
 namei: 245, 248
 ncursesw6-config: 162, 163
 neqn: 208, 209
 networkctl: 232, 235
 newgidmap: 151, 154
 newgrp: 151, 154
 newuidmap: 151, 154
 newusers: 151, 154
 ngettext: 167, 168
 nice: 201, 204
 ninja: 137, 137
 nl: 201, 204
 nm: 139, 140
 nohup: 201, 204
 nologin: 151, 154
 nproc: 201, 204
 nroff: 208, 209
 nsenter: 245, 248
 nstat: 216, 217
 numfmt: 201, 204
 objcopy: 139, 140
 objdump: 139, 140
 od: 201, 204
 oomctl: 232, 235
 openssl: 185, 186
 openvt: 218, 219
 partx: 245, 248
 passwd: 151, 154
 paste: 201, 204
 patch: 223, 223
 pathchk: 201, 204
 pcprofiledump: 109, 115
 pcre2grep: 127, 128
 pcre2test: 127, 128
 pdfmom: 208, 209
 pdffroff: 208, 209
 pdftexi2dvi: 225, 226
 peekfd: 166, 166
 perl: 180, 181
 perl5.44.0: 180, 181
 perlbug: 180, 181
 perldoc: 180, 181
 perlvp: 180, 181
 perlthanks: 180, 181
 pfbtops: 208, 209
 pgrep: 243, 243
 pic: 208, 209
 pic2graph: 208, 209
 piconv: 180, 181
 pidof: 243, 243
 ping: 177, 178
 ping6: 177, 178
 pinky: 201, 204
 pip3: 192
 pivot_root: 245, 248
 pkgconf: 138, 138
 pkill: 243, 243
 pl2pm: 180, 181
 pldd: 109, 115
 pmap: 243, 243
 pod2html: 180, 181
 pod2man: 180, 181
 pod2texi: 225, 226
 pod2text: 180, 181
 pod2usage: 180, 181
 podchecker: 180, 181
 podselect: 180, 181
 portablectl: 232, 235
 post-grohtml: 208, 209
 poweroff: 232, 235
 pr: 201, 204
 pre-grohtml: 208, 209
 preconv: 208, 209
 printenv: 201, 204
 printf: 201, 204
 prlimit: 245, 248
 prove: 180, 181
 prtstat: 166, 166
 ps: 243, 243
 psfaddtable: 218, 219
 psfgettable: 218, 219
 psfstripletable: 218, 219
 psfxtable: 218, 219
 pslog: 166, 166
 pstree: 166, 166
 pstree.x11: 166, 166
 ptar: 180, 181
 ptardiff: 180, 181
 ptargrep: 180, 182
 ptx: 201, 204
 pwck: 151, 154
 pwconv: 151, 154
 pwd: 201, 204

pwdx: 243, 244
 pwunconv: 151, 154
 pydoc3: 192
 python3: 192
 ranlib: 139, 141
 readelf: 139, 141
 readlink: 201, 204
 readprofile: 245, 248
 realpath: 201, 204
 reinicializar: 232, 235
 recode-sr-latin: 167, 168
 refer: 208, 210
 rename: 245, 248
 renice: 245, 248
 reset: 162, 163
 resize2fs: 250, 252
 resizepart: 245, 248
 resolvconf: 232, 235
 resolvectl: 232, 235
 rev: 245, 248
 rfkill: 245, 248
 rm: 201, 204
 rmdir: 201, 204
 rmmmod: 200, 200
 roff2dvi: 208, 210
 roff2html: 208, 210
 roff2pdf: 208, 210
 roff2ps: 208, 210
 roff2text: 208, 210
 roff2x: 208, 210
 routel: 216, 217
 rtacct: 216, 217
 rtcwake: 245, 248
 rtmon: 216, 217
 rtpr: 216, 217
 rtstat: 216, 217
 run0: 232, 235
 runlevel: 232, 235
 runtest: 136, 136
 rview: 227, 229
 rvim: 227, 229
 script: 245, 248
 scriptlive: 245, 248
 scriptreplay: 245, 248
 sdiff: 206, 206
 sed: 165, 165
 seq: 201, 204
 setarch: 245, 248
 setcap: 148, 148
 setfacl: 147, 147
 setfattr: 146, 146
 setfont: 218, 219
 setkeycodes: 218, 219
 setleds: 218, 219
 setmetamode: 218, 219
 setsid: 245, 248
 setterm: 245, 248
 setvtrgb: 218, 219
 sfdisk: 245, 248
 sg: 151, 154
 sh: 171, 172
 sha1sum: 201, 204
 sha224sum: 201, 204
 sha256sum: 201, 204
 sha384sum: 201, 204
 sha512sum: 201, 204
 shasum: 180, 182
 showconsolefont: 218, 219
 showkey: 218, 219
 shred: 201, 204
 shuf: 201, 204
 shutdown: 232, 235
 size: 139, 141
 slabtop: 243, 244
 sleep: 201, 204
 sln: 109, 115
 soelim: 208, 210
 sort: 201, 204
 sotruss: 109, 115
 splain: 180, 182
 split: 201, 204
 sprof: 109, 115
 sqlite3: 189, 189
 ss: 216, 217
 stat: 201, 205
 stdbuf: 201, 205
 strings: 139, 141
 strip: 139, 141
 stty: 201, 205
 su: 151, 154
 sulogin: 245, 248
 sum: 201, 205
 swaplabel: 245, 248
 swapoff: 245, 249
 swapon: 245, 249
 switch_root: 245, 249
 sync: 201, 205
 sysctl: 243, 244
 systemctl: 232, 235
 systemd-ac-power: 232, 235

systemd-analyze: 232, 235
 systemd-ask-password: 232, 235
 systemd-cat: 232, 235
 systemd-cgls: 232, 235
 systemd-cgtop: 232, 236
 systemd-creds: 232, 236
 systemd-delta: 232, 236
 systemd-detect-virt: 232, 236
 systemd-dissect: 232, 236
 systemd-escape: 232, 236
 systemd-hwdb: 232, 236
 systemd-id128: 232, 236
 systemd-inhibit: 232, 236
 systemd-machine-id-setup: 232, 236
 systemd-mount: 232, 236
 systemd-notify: 232, 236
 systemd-nspawn: 232, 236
 systemd-path: 232, 236
 systemd-pty-forward: 232, 236
 systemd-repart: 232, 236
 systemd-resolve: 232, 236
 systemd-run: 232, 236
 systemd-socket-activate: 232, 236
 systemd-sysex: 232, 236
 systemd-tmpfiles: 232, 236
 systemd-tty-ask-password-agent: 232, 237
 systemd-umount: 232, 236
 systemd-vpick: 232, 237
 tabs: 162, 163
 tac: 201, 205
 tail: 201, 205
 talk: 177, 178
 tar: 224, 224
 taskset: 245, 249
 tbl: 208, 210
 tc: 216, 217
 tclsh: 132, 133
 tclsh8.6: 132, 133
 tee: 201, 205
 telnet: 177, 178
 test: 201, 205
 texi2dvi: 225, 226
 texi2pdf: 225, 226
 texi2any: 225, 226
 texindex: 225, 226
 tfmtodit: 208, 210
 tftp: 177, 178
 tic: 162, 163
 timedatectl: 232, 237
 timeout: 201, 205
 tload: 243, 244
 toe: 162, 163
 top: 243, 244
 touch: 201, 205
 tput: 162, 163
 tr: 201, 205
 traceroute: 177, 178
 troff: 208, 210
 true: 201, 205
 truncate: 201, 205
 tset: 162, 163
 tsort: 201, 205
 tty: 201, 205
 tune2fs: 250, 252
 tzselect: 109, 115
 uclampset: 245, 249
 udevadm: 232, 237
 ul: 245, 249
 umount: 245, 249
 uname: 201, 205
 uname26: 245, 249
 uncompress: 215, 215
 unexpand: 201, 205
 unicode_start: 218, 219
 unicode_stop: 218, 220
 uniq: 201, 205
 unlink: 201, 205
 unlz4: 122, 122
 unlzma: 120, 121
 unshare: 245, 249
 unxz: 120, 121
 updatedb: 207, 207
 uptime: 243, 244
 useradd: 151, 154
 userdbctl: 232, 237
 userdel: 151, 154
 usermod: 151, 154
 users: 201, 205
 utmpdump: 245, 249
 uuidd: 245, 249
 uuidgen: 245, 249
 uuidparse: 245, 249
 varlinkctl: 232, 237
 vdir: 201, 205
 vi: 227, 229
 view: 227, 229
 vigr: 151, 154
 vim: 227, 229
 vimdiff: 227, 229
 vimtutor: 227, 229

vipw: 151, 154
 vmstat: 243, 244
 w: 243, 244
 wall: 245, 249
 watch: 243, 244
 wc: 201, 205
 wdctl: 245, 249
 whatis: 240, 242
 wheel: 197
 whereis: 245, 249
 who: 201, 205
 whoami: 201, 205
 wipefs: 245, 249
 x86_64: 245, 249
 xargs: 207, 207
 xgettext: 167, 168
 xmlwf: 176, 176
 xsubpp: 180, 182
 xtrace: 109, 115
 xxd: 227, 229
 xz: 120, 121
 xzcat: 120, 121
 xzcmp: 120, 121
 xzdec: 120, 121
 xzdiff: 120, 121
 xzegrep: 120, 121
 xzfgrep: 120, 121
 xzgrep: 120, 121
 xzless: 120, 121
 xzmore: 120, 121
 yacc: 169, 169
 yes: 201, 205
 zcat: 215, 215
 zcmp: 215, 215
 zdiff: 215, 215
 zdump: 109, 115
 zegrep: 215, 215
 zfgrep: 215, 215
 zforce: 215, 215
 zgrep: 215, 215
 zic: 109, 115
 zipdetails: 180, 182
 zless: 215, 215
 zmore: 215, 215
 znew: 215, 215
 zramctl: 245, 249
 zstd: 123, 123
 zstdgrep: 123, 123
 zstdless: 123, 123

ld-2.44.so: 109, 115
 libacl: 147, 147
 libanl: 109, 115
 libasprintf: 167, 168
 libattr: 146, 146
 libbfd: 139, 141
 libblkid: 245, 249
 libBrokenLocale: 109, 115
 libbz2: 118, 119
 libc: 109, 115
 libcap: 148, 148
 libcom_err: 250, 252
 libcrypt: 149, 150
 libcrypto.so: 185, 186
 libctf: 139, 141
 libctf-nobfd: 139, 141
 libc_malloc_debug: 109, 115
 libdbus-1: 238, 239
 libdl: 109, 115
 libe2p: 250, 252
 libelf: 187, 187
 libexpat: 176, 176
 libexpect-5.45.4: 134, 135
 libext2fs: 250, 252
 libfdisk: 245, 249
 libffi: 188
 libfl: 131, 131
 libformw: 162, 164
 libg: 109, 115
 libgcc: 156, 160
 libgcov: 156, 160
 libgdbm: 174, 174
 libgdbm_compat: 174, 174
 libgettextlib: 167, 168
 libgettextpo: 167, 168
 libgettextsrc: 167, 168
 libgmp: 142, 143
 libgmpxx: 142, 143
 libgomp: 156, 160
 libgprofng: 139, 141
 libhistory: 125, 126
 libhwasan: 156, 160
 libitm: 156, 160
 libkmod: 200
 liblsan: 156, 160
 libltdl: 173, 173
 liblto_plugin: 156, 160
 liblz4: 122, 122
 liblzma: 120, 121
 libm: 109, 115

libmagic: 124, 124
 libman: 240, 242
 libmandb: 240, 242
 libmcheck: 109, 116
 libmemusage: 109, 116
 libmenuw: 162, 164
 libmount: 245, 249
 libmpc: 145, 145
 libmpdec: 191, 191
 libmpdec++: 191, 191
 libmpfr: 144, 144
 libmvec: 109, 116
 libncurses++w: 162, 164
 libncursesw: 162, 164
 libnsl: 109, 116
 libnss_*: 109, 116
 libopcodes: 139, 141
 libpanelw: 162, 164
 libpcprofile: 109, 116
 libpipeline: 221
 libpkgconf: 138, 138
 libproc-2: 243, 244
 libpsx: 148, 148
 libpthread: 109, 116
 libquadmath: 156, 161
 libreadline: 125, 126
 libresolv: 109, 116
 librt: 109, 116
 libsframe: 139, 141
 libsmartcols: 245, 249
 libsqlite3.so: 189, 190
 libss: 250, 252
 libssl.so: 185, 186
 libssp: 156, 161
 libstdbuf: 201, 205
 libstdc++: 156, 161
 libstdc++exp: 156, 161
 libstdc++fs: 156, 161
 libsubid: 151, 154
 libsupc++: 156, 161
 libsystemd: 232, 237
 libtcl8.6.so: 132, 133
 libtclstub8.6.a: 132, 133
 libtextstyle: 167, 168
 libthread_db: 109, 116
 libtsan: 156, 161
 libubsan: 156, 161
 libudev: 232, 237
 libutil: 109, 116
 libuuid: 245, 249

liby: 169, 169
 libz: 117, 117
 libzstd: 123, 123
 preloadable_libintl: 167, 168

relógio
 configurando: 264
 console
 configurando: 265
 hostname
 configurando: 259
 rede local
 /etc/hosts: 259
 rede de comunicação
 /etc/hosts: 259
 configurando: 256
 rede de comunicação
 /etc/hosts: 259
 configurando: 256
 dwp: 139, 140

/boot/config-7.1.8: 277, 283
 /boot/System.map-7.1.8: 277, 284
 /dev/*: 84
 /etc/fstab: 275
 /etc/group: 87
 /etc/hosts: 259
 /etc/inputrc: 269
 /etc/ld.so.conf: 114
 /etc/lfs-release: 290
 /etc/localtime: 113
 /etc/lsb-release: 290
 /etc/mke2fs.conf: 251
 /etc/nsswitch.conf: 113
 /etc/os-release: 290
 /etc/passwd: 87
 /etc/profile: 267
 /etc/locale.conf: 267
 /etc/protocols: 108
 /etc/resolv.conf: 258
 /etc/services: 108
 /etc/vimrc: 228
 /run/utmp: 87
 /usr/include/asm-generic/*.h: 57, 57
 /usr/include/asm/*.h: 57, 57
 /usr/include/drm/*.h: 57, 57
 /usr/include/linux/*.h: 57, 57
 /usr/include/misc/*.h: 57, 57
 /usr/include/mtd/*.h: 57, 57
 /usr/include/rdma/*.h: 57, 57

/usr/include/scsi/*.h: 57, 57
/usr/include/sound/*.h: 57, 57
/usr/include/video/*.h: 57, 57
/usr/include/xen/*.h: 57, 57
/var/log/btmp: 87
/var/log/lastlog: 87
/var/log/wtmp: 87
/etc/shells: 270
páginas de manual: 107, 107
Personalização do Systemd: 271