

Linux From Scratch

Versão 12.2

Publicado 1º de setembro de 2024

**Criado por Gerard Beekmans
Editor-chefe: Bruce Dubbs**

Linux From Scratch: Versão 12.2: Publicado 1º de setembro de 2024

por Criado por Gerard Beekmans e Editor-chefe: Bruce Dubbs

Copyright © 1999-2024 Gerard Beekmans

Direitos autorais © 1999-2024, 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. Introdução	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	xi
vii. Tipografia	xvii
viii. Estrutura	xviii
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	11
2. Preparando o Sistema Anfitrião	12
2.1. Introdução	12
2.2. Exigências do Sistema Anfitrião	12
2.3. Construindo o LFS em Estágios	14
2.4. Criando uma Nova Partição	15
2.5. Criando um Sistema de Arquivos na Partição	17
2.6. Configurando a Variável \$LFS	18
2.7. Montando a Nova Partição	19
3. Pacotes e Remendos	21
3.1. Introdução	21
3.2. Todos os Pacotes	22
3.3. Remendos Necessários	30
4. Preparações Finais	32
4.1. Introdução	32
4.2. Criando um Layout Limitado de Diretório no Sistema de Arquivos do LFS	32
4.3. Adicionando o(a) Usuário(a) LFS	32
4.4. Configurando o Ambiente	33
4.5. A Respeito de UPCs	36
4.6. A Respeito das Suítes de Teste	37
III. Construindo o Conjunto de Ferramentas Cruzadas do LFS e Ferramentas Temporárias	38
Material Preliminar Importante	xxxix
i. Introdução	xxxix
ii. Observações Técnicas do Conjunto de Ferramentas	xxxix
iii. Instruções Gerais de Compilação	xliv
5. Compilando um Conjunto de Ferramentas Cruzado	46
5.1. Introdução	46
5.2. Binutils-2.43.1 - Passagem 1	47
5.3. GCC-14.2.0 - Passagem 1	49
5.4. Cabeçalhos da API do Linux-6.10.5	52
5.5. Glibc-2.40	53

5.6. Libstdc++ oriundo de GCC-14.2.0	56
6. Compilando Cruzadamente Ferramentas Temporárias	58
6.1. Introdução	58
6.2. M4-1.4.19	59
6.3. Ncurses-6.5	60
6.4. Bash-5.2.32	62
6.5. Coreutils-9.5	63
6.6. Diffutils-3.10	64
6.7. File-5.45	65
6.8. Findutils-4.10.0	66
6.9. Gawk-5.3.0	67
6.10. Grep-3.11	68
6.11. Gzip-1.13	69
6.12. Make-4.4.1	70
6.13. Patch-2.7.6	71
6.14. Sed-4.9	72
6.15. Tar-1.35	73
6.16. Xz-5.6.2	74
6.17. Binutils-2.43.1 - Passagem 2	75
6.18. GCC-14.2.0 - Passagem 2	76
7. Entrando no Chroot e Construindo Ferramentas Temporárias Adicionais	78
7.1. Introdução	78
7.2. Mudando Propriedade	78
7.3. Preparando Sistemas de Arquivos Virtuais do Núcleo	78
7.4. Entrando no Ambiente Chroot	80
7.5. Criando Diretórios	80
7.6. Criando Arquivos Essenciais e Links Simbólicos	81
7.7. Gettext-0.22.5	84
7.8. Bison-3.8.2	85
7.9. Perl-5.40.0	86
7.10. Python-3.12.5	87
7.11. Texinfo-7.1	88
7.12. Util-linux-2.40.2	89
7.13. Limpando e Salvando o Sistema Temporário	90
IV. Construindo o Sistema LFS	92
8. Instalando Aplicativos Básicos de Sistema	93
8.1. Introdução	93
8.2. Gerenciamento de Pacote	94
8.3. Man-pages-6.9.1	99
8.4. Iana-Etc-20240806	100
8.5. Glibc-2.40	101
8.6. Zlib-1.3.1	109
8.7. Bzip2-1.0.8	110
8.8. Xz-5.6.2	112
8.9. Lz4-1.10.0	114
8.10. Zstd-1.5.6	115
8.11. File-5.45	116
8.12. Readline-8.2.13	117
8.13. M4-1.4.19	119
8.14. Bc-6.7.6	120

8.15. Flex-2.6.4	121
8.16. Tcl-8.6.14	122
8.17. Expect-5.45.4	124
8.18. DejaGNU-1.6.3	126
8.19. Pkgconf-2.3.0	127
8.20. Binutils-2.43.1	128
8.21. GMP-6.3.0	131
8.22. MPFR-4.2.1	133
8.23. MPC-1.3.1	134
8.24. Attr-2.5.2	135
8.25. Acl-2.3.2	136
8.26. Libcap-2.70	137
8.27. Libxcrypt-4.4.36	138
8.28. Shadow-4.16.0	140
8.29. GCC-14.2.0	144
8.30. Ncurses-6.5	150
8.31. Sed-4.9	153
8.32. Psmisc-23.7	154
8.33. Gettext-0.22.5	155
8.34. Bison-3.8.2	157
8.35. Grep-3.11	158
8.36. Bash-5.2.32	159
8.37. Libtool-2.4.7	161
8.38. GDBM-1.24	162
8.39. Gperf-3.1	163
8.40. Expat-2.6.2	164
8.41. Inetutils-2.5	165
8.42. Less-661	167
8.43. Perl-5.40.0	168
8.44. XML::Parser-2.47	171
8.45. Intltool-0.51.0	172
8.46. Autoconf-2.72	173
8.47. Automake-1.17	174
8.48. OpenSSL-3.3.1	175
8.49. Kmod-33	177
8.50. Libelf originário do Elfutils-0.191	179
8.51. Libffi-3.4.6	180
8.52. Python-3.12.5	181
8.53. Flit-Core-3.9.0	183
8.54. Wheel-0.44.0	184
8.55. Setuptools-72.2.0	185
8.56. Ninja-1.12.1	186
8.57. Meson-1.5.1	187
8.58. Coreutils-9.5	188
8.59. Check-0.15.2	193
8.60. Diffutils-3.10	194
8.61. Gawk-5.3.0	195
8.62. Findutils-4.10.0	196
8.63. Groff-1.23.0	197
8.64. GRUB-2.12	200

8.65. Gzip-1.13	202
8.66. IPRoute2-6.10.0	203
8.67. Kbd-2.6.4	205
8.68. Libpipeline-1.5.7	208
8.69. Make-4.4.1	209
8.70. Patch-2.7.6	210
8.71. Tar-1.35	211
8.72. Texinfo-7.1	212
8.73. Vim-9.1.0660	214
8.74. MarkupSafe-2.1.5	217
8.75. Jinja2-3.1.4	218
8.76. Udev originário de Systemd-256.4	219
8.77. Man-DB-2.12.1	222
8.78. Procps-ng-4.0.4	225
8.79. Util-linux-2.40.2	227
8.80. E2fsprogs-1.47.1	232
8.81. Sysklogd-2.6.1	235
8.82. SysVinit-3.10	236
8.83. Acerca dos Símbolos de Depuração	237
8.84. Despojando	237
8.85. Limpando	239
9. Configuração do Sistema	240
9.1. Introdução	240
9.2. LFS-Bootscripts-20240825	241
9.3. Visão Geral do Manuseio de Dispositivo e de Módulo	243
9.4. Gerenciando Dispositivos	246
9.5. Configuração Geral da Rede de Comunicação	249
9.6. Uso e Configuração do Script de Inicialização do System V	252
9.7. Configurando a Localidade do Sistema	261
9.8. Criando o Arquivo /etc/inputrc	262
9.9. Criando o Arquivo /etc/shells	263
10. Tornando o Sistema LFS Inicializável	265
10.1. Introdução	265
10.2. Criando o Arquivo /etc/fstab	265
10.3. Linux-6.10.5	267
10.4. Usando o GRUB para Configurar o Processo de Inicialização	273
11. O Fim	276
11.1. O Fim	276
11.2. Seja Contado(a)	276
11.3. Reinicializando o Sistema	276
11.4. Recursos Adicionais	277
11.5. Começando Depois do LFS	278
V. Anexos	281
A. Siglas e Termos	282
B. Reconhecimentos	284
C. Dependências	287
D. Scripts de inicialização e configuração do sistema versão-20240825	307
D.1. /etc/rc.d/init.d/rc	307
D.2. /lib/lsb/init-functions	310
D.3. /etc/rc.d/init.d/mountvirtfs	323

D.4. /etc/rc.d/init.d/modules	325
D.5. /etc/rc.d/init.d/udev	326
D.6. /etc/rc.d/init.d/swap	327
D.7. /etc/rc.d/init.d/setclock	328
D.8. /etc/rc.d/init.d/checkfs	329
D.9. /etc/rc.d/init.d/mountfs	331
D.10. /etc/rc.d/init.d/udev_retry	333
D.11. /etc/rc.d/init.d/cleanfs	334
D.12. /etc/rc.d/init.d/console	336
D.13. /etc/rc.d/init.d/localnet	337
D.14. /etc/rc.d/init.d/sysctl	338
D.15. /etc/rc.d/init.d/syslogd	339
D.16. /etc/rc.d/init.d/network	340
D.17. /etc/rc.d/init.d/sendsignals	342
D.18. /etc/rc.d/init.d/reboot	343
D.19. /etc/rc.d/init.d/halt	344
D.20. /etc/rc.d/init.d/template	344
D.21. /etc/sysconfig/modules	345
D.22. /etc/sysconfig/createfiles	346
D.23. /etc/sysconfig/udev-retry	346
D.24. /sbin/ifup	347
D.25. /sbin/ifdown	349
D.26. /lib/services/ipv4-static	350
D.27. /lib/services/ipv4-static-route	352
E. Regras de configuração do Udev	354
E.1. 55-lfs.rules	354
F. Licenças do LFS	355
F.1. Licença da Creative Commons	355
F.2. A Licença do MIT	359
Índice Remissivo	360

Prefácio

Introdução

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 utilizável, 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 dos(as) usuários(as), 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 seu 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 para adaptá-las às suas necessidades, 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 sua própria jornada.

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

--

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 seu sistema.

O LFS te permite criar sistemas Linux muito compactos. Com outras distribuições, você frequentemente é forçado(a) a instalar grande número de aplicativos, os quais nem usa, nem entende. Esses aplicativos 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.

A objetivo do Linux From Scratch é o de construir um sistema em nível de fundação completo e utilizável. Se você não deseja construir seu próprio sistema Linux desde o zero, [então] você possivelmente nunca se beneficie das informações neste livro.

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

Arquiteturas Alvo do LFS

As arquiteturas alvo primárias do LFS são as CPUs AMD/Intel x86 (32 bits) e x86_64 (64 bits). Por outro lado, as instruções neste livro também são conhecidas por funcionar, com algumas modificações, com as CPUs Power PC e ARM. Para construir um sistema que utiliza uma dessas CPUs alternativas, o principal pré-requisito, em adição àqueles na próxima página, é um sistema Linux existente, como uma instalação prévia do LFS, Ubuntu, Red Hat/Fedora, SuSE ou alguma outra distribuição que atinja essa arquitetura. (Observe que uma distribuição de 32 bits pode ser instalada e usada como um sistema anfitrião em um computador AMD/Intel de 64 bits).

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% mais larga) que a construção de 32 bits. Se você planeja usar o LFS como um servidor LAMP, ou como um firewall, [então] 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, [então] os(as) autores(as) do LFS recomendam construir um sistema de 64 bits.

A construção padrão de 64 bits que resulta do LFS é um sistema de 64 bits “puro”. Ou seja, ele suporta somente executáveis de 64 bits. Construir um sistema “multi biblioteca” exige compilar muitos aplicativos duas vezes, uma vez para um sistema de 32 bits e outra vez para um sistema de 64 bits. Isso não é diretamente suportado no LFS, pois interferiria no objetivo educacional de fornecer as instruções mínimas necessárias para um sistema básico Linux. Alguns(mas) dos(as) editores(as) do LFS/BLFS mantém uma bifurcação multi biblioteca do LFS, acessível em <https://www.linuxfromscratch.org/~thomas/multilib/index.html>. Porém, esse é um tópico avançado.

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 *pele 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) Version 3.0*
- *Linux Standard Base (LSB) Version 5.0 (2015)*

O LSB tem quatro especificações separadas: Core, Desktop, Runtime 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 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) usuário(a) tem total controle sobre quais pacotes são desejados; você possivelmente escolha não instalar alguns pacotes que são especificados pelo LSB.

Ao tempo em que é possível criar um sistema completo que passará nos testes de certificação do "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 esses 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, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux e Zlib
<i>Área de trabalho do LSB:</i>	Nenhum
<i>Linguagens de Tempo de Execução do LSB:</i>	Perl e Python
<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), Cpio, Ed, Fcfrontab, LSB-Tools, NSPR, NSS, PAM, Pax, Sendmail (ou Postfix ou Exim), time
<i>Área de trabalho do LSB:</i>	Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GTK+2, Icon-naming-utils, Libjpeg-turbo, Libpng, Libtiff, Libxml2, MesaLib, Pango, Xdg-utils e Xorg
<i>Linguagens de Tempo de Execução do 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

Pacotes não fornecidos pelo LFS ou pelo BLFS necessários para satisfazer as Exigências do LSB

<i>Núcleo do LSB:</i>	Nenhum
<i>Área de trabalho do LSB:</i>	Qt4 (mas Qt5 é fornecido)
<i>Linguagens de Tempo de Execução do LSB:</i>	Nenhum
<i>Imagem no LSB:</i>	Nenhum
<i>LSB Gtk3 e Gráficos LSB (Uso Experimental):</i>	Nenhum

Justificativa para os Pacotes no Livro

O objetivo do LFS é o de construir um sistema em nível de fundação completo e utilizável—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) usuário(a). 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 as razões 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.

- Check

Esse pacote fornece um equipamento de teste para outros aplicativos.

- 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.

- 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 Udev.
- 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.

- Intltool

Esse pacote contribui com ferramentas para extrair sequências de caracteres traduzíveis a partir de arquivos fonte.

- 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.

- 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(a) 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 `libxcrypt` 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.

- 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.

- 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.

- 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.
- Sysklogd

Esse pacote fornece aplicativos para registrar mensagens do sistema, tais como aquelas emitidas pelo núcleo ou por processos deemons quando eventos não-usuais acontecem.
- SysVinit

Esse pacote fornece o aplicativo init, o antepassado de todos os outros processos em um Linux em execução.
- Udev

Esse pacote é um gerenciador de dispositivo. Controla dinamicamente a titularidade da propriedade, permissões, nomes e links simbólicos de nós de dispositivos no diretório /dev quando dispositivos são adicionados ou removidos do sistema.
- 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 muitas(os) usuárias(os). 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.

- XML::Parser

Esse pacote é um módulo do Perl que interage com o Expat.

- 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. Esse formato 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 `ll` ou `oo`.

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 usa “*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 tem nomes duplicados de páginas em diferentes seções. Portanto, “**man <nome do aplicativo>**” geralmente é suficiente. No livro LFS essas referências para páginas de manual também são hiperlinks, de forma 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/12.2/> 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/lfs/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.

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:

-

- Automake-1.17
- Bash-5.2.32
- Bc-6.7.6
- Binutils-2.43.1
- Coreutils-9.5
- E2fsprogs-1.47.1
- Expat-2.6.2
- Findutils-4.10.0
- GCC-14.2.0
- GDBM-1.24
- Gettext-0.22.5
- Glibc-2.40
- Iana-Etc-20240806
- IPRoute2-6.10.0
- Jinja2-3.1.4
- Kmod-33
- Less-661
- Libcap-2.70
- Libelf originário do Elfutils-0.191
- Libffi-3.4.6
- Linux-6.10.5
- Lz4-1.10.0
- Man-DB-2.12.1
- Man-pages-6.9.1
- Meson-1.5.1
- Ncurses-6.5
- Ninja-1.12.1
- OpenSSL-3.3.1
- Perl-5.40.0
- Pkgconf-2.3.0
- Psmisc-23.7
- Python-3.12.5
- Readline-8.2.13
- Setuptools-72.2.0
- Shadow-4.16.0
- Sysklogd-2.6.1
- Systemd-256.4

- SysVinit-3.10
- Tcl-8.6.14
- Udev originário de Systemd-256.4
- Util-linux-2.40.2
- Vim-9.1.0660
- Wheel-0.44.0
- Xz-5.6.2
- Zstd-1.5.6

Adicionado:

- Lz4-1.10.0

Removido:

- bash-5.2.21-upstream_fixes-1.patch
- readline-8.2-upstream_fixes-3.patch

1.3. Registro das Mudanças

Esta é a versão 12.2 do livro Linux From Scratch, datada de 1º de setembro de 2024. 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:

- 2024-09-01
 - [bdubbs] - LFS-12.2 lançado.
- 2024-08-23
 - [xry111] - Atualização para lfs-bootscripsts-20240825. Somente mudanças triviais não funcionais.
- 2024-08-23
 - [xry111] - Atualização para lfs-bootscripsts-20240823, para corrigir um problema causante de Terminal Virtual 2-6 não afetados pela configuração FONT= em /etc/sysconfig/console.
- 2024-08-17
 - [bdubbs] - Atualizado para setuptools-72.2.0. Corrige #5542.
 - [bdubbs] - Atualizado para kmod-33. Corrige #5540.
 - [bdubbs] - Atualizado para binutils-2.43.1. Corrige #5543.
 - [bdubbs] - Atualizado para linux-6.10.5. Corrige #5541.
- 2024-08-15
 - [bdubbs] - Atualização para iana-etc-20240806. Endereça #5006.
 - [bdubbs] - Atualizado para pkgconf-2.3.0. Corrige #5537.
 - [bdubbs] - Atualizado para python3-3.12.5. Corrige #5538.
 - [bdubbs] - Atualizado para linux-6.10.4. Corrige #5539.
- 2024-08-05

- [bdubbs] - Atualizado para bash-5.2.32. Corrige #5532.
- [bdubbs] - Atualização para iana-etc-20240801. Endereça #5006.
- [bdubbs] - Atualização para to vim-9.1.0660. Endereça #4500.
- [bdubbs] - Atualizado para binutils-2.43. Corrige #5535.
- [bdubbs] - Atualizado para linux-6.10.3. Corrige #5534.
- [bdubbs] - Atualizado para readline-8.2.13. Corrige #5533.
- [bdubbs] - Atualizado para wheel-0.44.0. Corrige #5536.
- 2024-08-01
 - [bdubbs] - Atualizado para gcc-14.2.0. Corrige #5530.
 - [bdubbs] - Atualização para iana-etc-20240723. Endereça #5006.
 - [bdubbs] - Atualizado para glibc-2.40. Corrige #5529.
 - [bdubbs] - Atualizado para iproute2-6.10.0. Corrige #5523.
 - [bdubbs] - Atualizado para linux-6.10.2. Corrige #5521.
 - [bdubbs] - Atualizado para lz4-1.10.0. Corrige #5526.
 - [bdubbs] - Atualizado para meson-1.5.1. Corrige #5527.
 - [bdubbs] - Atualizado para setuptools-72.1.0. Corrige #5531.
 - [bdubbs] - Atualizado para syslogd-2.6.1. Corrige #5522.
 - [bdubbs] - Atualizado para systemd-256.4. Corrige #5518.
 - [bdubbs] - Atualizado para sysvinit-3.10. Corrige #5528.
- 2024-07-15
 - [bdubbs] - Atualização para iana-etc-20240701. Endereça #5006.
 - [bdubbs] - Atualização para vim-9.1.0580. Endereça #4500.
 - [bdubbs] - Atualizado para automake-1.17. Corrige #5520.
 - [bdubbs] - Atualizado para gdbm-1.24. Corrige #5515.
 - [bdubbs] - Atualizado para linux-6.9.9. Corrige #5517.
 - [bdubbs] - Atualizado para less-661. Corrige #5513.
 - [bdubbs] - Atualizado para meson-1.5.0. Corrige #5519.
 - [bdubbs] - Atualizado para setuptools-70.3.0. Corrige #5514.
 - [bdubbs] - Atualizado para util-linux-2.40.2. Corrige #5516.
- 2024-07-01
 - [bdubbs] - Atualizar lfs-bootscripsts para gerar somente sequências de escape para um terminal.
- 2024-07-01
 - [bdubbs] - Atualização para iana-etc-20240612. Endereça #5006.
 - [bdubbs] - Atualizado para bc-6.7.6. Corrige #5506.
 - [bdubbs] - Atualizado para man-pages-6.9.1. Corrige #5507.
 - [bdubbs] - Atualizado para linux-6.9.7. Corrige #5508.
 - [bdubbs] - Atualizado para syslogd-2.5.2. Corrige #5509.
 - [bdubbs] - Atualizado para shadow-4.16.0. Corrige #5510.

- [bdubbs] - Atualizado para systemd-256.1. Corrige #5511.
- [bdubbs] - Atualizado para setuputils-70.1.1. Corrige #5512.
- 2024-06-15
 - [bdubbs] - Atualização para vim-9.1.0478. Endereça #4500.
 - [bdubbs] - Atualização para iana-etc-20240607. Endereça #5006.
 - [bdubbs] - Atualizado para systemd-256. Corrige #5504.
 - [bdubbs] - Atualizado para python3-3.12.4. Corrige #5502.
 - [bdubbs] - Atualizado para perl-5.40.0. Corrige #5503.
 - [bdubbs] - Atualização para openssl-3.3.1 (correção de segurança). Corrige #5500.
 - [bdubbs] - Atualizado para linux-6.9.4. Corrige #5505.
 - [bdubbs] - Atualizado para findutils-4.10.0. Corrige #5499.
- 2024-05-31
 - [bdubbs] - Atualizado para meson-1.4.1. Corrige #5498.
 - [bdubbs] - Atualizado para xz-5.6.2. Corrige #5471.
 - [bdubbs] - Adicionar instruções de compatibilidade do linux-6.9.x ao udev. Corrige #5496.
 - [bdubbs] - Atualizado para setuputils-70.0.0 (módulo python). Corrige #5491.
 - [bdubbs] - Atualizado para ninja-1.12.1. Corrige #5489.
 - [bdubbs] - Atualizado para man-pages-6.8. Corrige #5494.
 - [bdubbs] - Atualizado para linux-6.9.3. Corrige #5491.
 - [bdubbs] - Atualizado para libcap-2.70. Corrige #5493.
 - [bdubbs] - Atualizado para iproute2-6.9.0. Corrige #5492.
 - [bdubbs] - Atualizado para e2fsprogs-1.47.1. Corrige #5495.
- 2024-05-13
 - [xry111] - Sincronizar o remendo i18n do coreutils oriundo do Fedora para corrigir uma falha de construção no x86 de 32 bits e um aviso alarmante do compilador em ambos, x86 de 32 bits e x86_64, com GCC 14.1 ou posterior.
- 2024-05-11
 - [bdubbs] - Atualização para vim-9.1.0405. Endereça #4500.
 - [bdubbs] - Atualizado para util-linux-2.40.1. Corrige #5482.
 - [bdubbs] - Atualizado para linux-6.8.9. Corrige #5484.
 - [bdubbs] - Atualizado para jinja2-3.1.4 (módulo Python). Corrige #5485.
 - [bdubbs] - Atualização para iana-etc-20240502. Endereça #5006.
 - [bdubbs] - Atualizado para gcc-14.1.0. Corrige #5486.
- 2024-05-01
 - [bdubbs] - Adicionar correção de segurança para glibc. Corrige #5481.
 - [bdubbs] - Atualizado para linux-6.8.8. Corrige #5480.
 - [bdubbs] - Atualizado para ncurses-6.5. Corrige #5483.
- 2024-04-16
 - [bdubbs] - Fazer pequenas mudanças no texto da saída gerada do conjunto de comandos sequenciais ifup.

- 2024-04-15
 - [bdubbs] - Atualizado para setuptools-69.5.1. Corrige #5478.
 - [bdubbs] - Atualizado para python3-3.12.3. Corrige #5476.
 - [bdubbs] - Atualizado para openssl-3.3.0. Corrige #5475.
 - [bdubbs] - Atualizado para ninja-1.12.0. Corrige #5477.
 - [bdubbs] - Atualizado para man-db-2.12.1. Corrige #5474.
 - [bdubbs] - Atualizado para linux-6.8.6. Corrige #5472.
 - [bdubbs] - Atualização para iana-etc-20240412. Endereça #5006.
 - [bdubbs] - Atualização para vim-9.1.0330. Endereça #4500.
- 2024-03-31
 - [bdubbs] - Atualização para iana-etc-20240318. Endereça #5006.
 - [bdubbs] - Atualizado para zstd-1.5.6. Corrige #5468.
 - [bdubbs] - Atualizado para util-linux-2.40. Corrige #5430.
 - [bdubbs] - Atualizado para shadow-4.15.1. Corrige #5465.
 - [bdubbs] - Atualizado para pkgconf-2.2.0. Corrige #5469.
 - [bdubbs] - Atualizado para linux-6.8.2. Corrige #5467.
 - [bdubbs] - Atualizado para coreutils-9.5. Corrige #5470.
- 2024-03-29
 - [bdubbs] - Reverter para xz-5.4.6 devido a comprometimento do fluxo de desenvolvimento.
- 2024-03-25
 - [timtas] - Atualizado para sysvinit-3.09. Corrige #5466.
- 2024-03-19
 - [renodr] - Atualizado para iproute2-6.8.0. Corrige #5458.
 - [renodr] - Atualizado para man-pages-6.7. Corrige #5464.
 - [renodr] - Atualizado para Linux-6.8.1. Corrige #5453.
 - [renodr] - Adicionado LZ4 ao livro. Corrige #5463.
- 2024-03-15
 - [bdubbs] - Atualizado para wheel-0.43.0. Corrige #5459.
 - [bdubbs] - Atualizado para setuptools-69.2.0 (módulo Python). Corrige #5462.
 - [bdubbs] - Atualizado para meson-1.4.0. Corrige #5460.
 - [bdubbs] - Atualização para expat-2.6.2 (Correção de segurança). Corrige #5461.
 - [bdubbs] - Atualização para iana-etc-20240305. Endereça #5006.
 - [bdubbs] - Atualização para vim-9.1.0161. Endereça #4500.
 - [bdubbs] - Atualizado para xz-5.6.1. Corrige #5457.
 - [bdubbs] - Atualizado para shadow-4.15.0. Corrige #5456.
 - [bdubbs] - Atualizado para psmisc-23.7. Corrige #5454.
 - [bdubbs] - Atualizado para kmod-32. Corrige #5455.
 - [bdubbs] - Atualizado para elfutils-0.191. Corrige #5451.

- 2024-03-02
 - [bdubbs] - Atualização para iana-etc-20240222. Endereça #5006.
 - [bdubbs] - Atualização para vim-9.1.0145. Endereça #4500.
 - [bdubbs] - Atualizado para xz-5.6.0. Corrige #5447.
 - [bdubbs] - Atualizado para tcl-8.6.14. Corrige #5448.
 - [bdubbs] - Atualizado para shadow-4.14.6. Corrige #5450.
 - [bdubbs] - Atualizado para setuptools-69.1.1. Corrige #5446.
 - [bdubbs] - Atualizado para linux-6.7.7. Corrige #5444.
 - [bdubbs] - Atualizado para libffi-3.4.6. Corrige #5443.
 - [bdubbs] - Atualizado para gettext-0.22.5. Corrige #5445.
 - [bdubbs] - Atualizado para expat-2.6.1. Corrige #5449.
- 2024-03-01
 - [bdubbs] - LFS-12.1 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` 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`. 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 12.2)
- 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.

- **Bash-3.2** (/bin/sh deveria ser um link simbólico ou real para bash)
- **Binutils-2.13.1** (Versões maiores que 2.43.1 não são recomendadas dado que elas não foram testadas)
- **Bison-2.7** (/usr/bin/yacc deveria ser um link para bison ou script pequeno que executa bison)
- **Coreutils-8.1**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk deveria ser um link para gawk)
- **GCC-5.2** incluindo o compilador C++, **g++** (Versões maiores que 14.2.0 não são recomendadas dado que elas não foram testadas). As bibliotecas C e C++ padrão (com cabeçalhos) também precisam estar presentes, de forma que o compilador C++ possa construir aplicativos hospedados
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Núcleo Linux-4.19**

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 fevereiro de 2024, 4.19 é o lançamento mais antigo do núcleo ainda suportado pelos(as) desenvolvedores(as) do núcleo. Alguns lançamentos de núcleo mais antigos que 4.19 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 4.19, [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 4.19 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 4.19 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**



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
# Um conjunto sequencial de comandos para listar os números de versão de ferramentas críticas de desenvolvimento

# Se você tiver ferramentas instaladas em outros diretórios, [então] ajuste "PATH" aqui E
# em ~/.bashrc (seção 4.4) também.

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

bail() { echo "FATAL: $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep não funciona"
sed '' /dev/null || bail "sed não funciona"
sort /dev/null || bail "sort não funciona"

ver_check()
{
    if ! type -p $2 &>/dev/null
    then
        echo "ERRO: Não conseguiu encontrar $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 "ERRO:  %-9s é MUITO ANTIGO ($3 ou posterior exigido)\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:    núcleo Linux $kver >= $1\n"; return 0;
else
    printf "ERRO:  núcleo Linux ($kver) é MUITO ANTIGO ($1 ou posterior exigido)\n" "$kver";
    return 1;
fi
}

# "Coreutils" primeiro, pois "--version-sort" precisa do "Coreutils" >= 7.0
ver_check Coreutils    sort      8.1 || bail "Coreutils muito antigo, pare"
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      5.2
ver_check "GCC (C++)"  g++     5.2
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 4.19

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK:    núcleo Linux suporta PTY do UNIX 98";
else echo "ERRO:  núcleo Linux NÃO suporta PTY do UNIX 98"; fi

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

echo "Verificação do compilador:"
if printf "int main(){}" | g++ -x c++ -
then echo "OK:    g++ funciona";
else echo "ERRO:  g++ NÃO funciona"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
    echo "ERRO:  nproc não está disponível ou ele produz saída gerada vazia"
else
    echo "OK:    nproc informa que $(nproc) núcleos lógicos estão disponíveis"
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) usuário(a) `root` após a Seção 2.4 precisam ter a variável de ambiente `LFS` configurada *PARA A(O) USUÁRIA(O) ROOT*.

2.3.2. Capítulos 5–6

- A partição `/mnt/lfs` precisa estar montada.
- Esses dois capítulos *precisam* ser feitos como o(a) usuário(a) `lfs`. Um comando **su - lfs** precisa ser emitido antes de realizar qualquer tarefa nesses capítulos. Se você não fizer isso, [então] você está no risco de instalar pacotes no sistema anfitrião e potencialmente torná-lo inutilizável.
- 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.
- Umhas poucas operações, desde “Mudando o(a) Dono(a)” até “Entrando no Ambiente Chroot”, precisam ser feitas como o(a) usuário(a) `root`, com a variável de ambiente `LFS` configurada para o(a) usuário(a) `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.

Um sistema mínimo exige uma partição com cerca de dez (10) gigabytes (GB). Isso é suficiente para armazenar todos os tarballs dos fontes e compilar os pacotes. Entretanto, se o sistema `LFS` for concebido para ser o sistema Linux principal, [então] aplicativos adicionais provavelmente serão instalados os quais exigirão espaço adicional. Uma partição de trinta (30) GB é um tamanho razoável para permitir o crescimento. O sistema `LFS` em si não ocupará esse espaço todo. Uma boa parte dessa exigência é para fornecer armazenamento temporário livre suficiente. Adicionalmente, a compilação de pacotes pode exigir um monte de espaço de disco que será recuperado após o pacote ser instalado.

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 `cfdisk` 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 `cfdisk(8)` ou a `fdisk(8)` se você ainda não sabe como usar os aplicativos.



Nota

Para usuários experientes, outros esquemas de partição 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 usuárias(os) do LFS pela primeira vez.

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 Grub

Se o *disco de inicialização* tiver sido particionado com a Tabela de Partição GUID (GPT), então uma partição pequena, tipicamente um (1) MB, precisa ser criada se 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**.



Nota

A Partição de Bios 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 Grub depende apenas do tipo de tabela de partição do disco de inicialização.

2.4.1.4. 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.
- `/boot/efi` – A Partição do Sistema EFI, a qual é necessária para inicializar o sistema com UEFI. Leia-se a *página do BLFS* para detalhes.
- `/home` – Altamente recomendada. Compartilhe seu diretório home e personalizações de usuário(a) 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` – Uma partição separada `/tmp` é rara, mas útil ao se configurar um "thin client". Essa partição, se usada, geralmente não precisará exceder um par de gigabytes. Se você tiver RAM suficiente, você consegue montar um `tmpfs` no `/tmp` para tornar o acesso a arquivos temporários mais rápido.
- `/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`".

2.5. 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 rastreamento 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 swap existente, [então] não há necessidade de formatá-la. Se uma nova partição swap foi criada, [então] ela precisará ser inicializada com este comando:

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

Substitua <yyy> pelo nome da partição swap.

2.6. Configurando a Variável \$LFS

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, [então] 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.



Cuidado

Não se esqueça de verificar se LFS está configurada sempre que você deixar e entrar novamente no ambiente atual de trabalho (como quando fizer um **su** para root ou outro(a) usuário(a)). Verifique se a variável LFS está configurada apropriadamente com:

```
echo $LFS
```

Tenha certeza de que a saída gerada mostra o caminho para o seu local de construção do sistema LFS, o qual é /mnt/lfs se o exemplo fornecido foi seguido. Se a saída gerada estiver incorreta, [então] use o comando dado anteriormente nesta página para configurar \$LFS para o nome correto de diretório.



Nota

Uma maneira de assegurar que a variável `LFS` sempre esteja configurada é a de editar o arquivo `.bash_profile` tanto em seu diretório `home` pessoal quanto em `/root/.bash_profile` e inserir o comando `export` acima. Adicionalmente, o shell especificado no arquivo `/etc/passwd` para todas(os) as(os) usuárias(os) que precisam da variável `LFS` precisa ser o `bash` para assegurar que o arquivo `/root/.bash_profile` seja incorporado como parte do processo de login.

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

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



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. Um tarball de lançamento contém arquivos generalizados (por exemplo, um script **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>.

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 usuária(o) `root`, antes de começar a sessão de transferência:

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

Torne esse diretório gravável e "sticky". "Sticky" significa que, mesmo se múltiplas(os) usuárias(os) tenham permissão de escrita, só a(o) dona(o) de um arquivo pode 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-sysv` como uma entrada gerada para o comando **wget**, use:

```
wget --input-file=wget-list-sysv --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) usuário(a) não `root`, [então] esses arquivos serão de propriedade do(a) usuário(a). O sistema de arquivos registra o(a) proprietário(a) pelo UID dele(a) e o UID de um(a) usuário(a) 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 o(a) seu(u) usuário(a) no sistema LFS, [então] mude os 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 (2.3.2) - 363 KB:**

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

Transferência: <https://download.savannah.gnu.org/releases/acl/acl-2.3.2.tar.xz>

Soma de verificação MD5: 590765dee95907dbc3c856f7255bd669

- **Attr (2.5.2) - 484 KB:**

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

Transferência: <https://download.savannah.gnu.org/releases/attr/attr-2.5.2.tar.gz>

Soma de verificação MD5: 227043ec2f6ca03c0948df5517f9c927

- **Autoconf (2.72) - 1.360 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/autoconf/autoconf-2.72.tar.xz>

Soma de verificação MD5: 1be79f7106ab6767f18391c5e22be701

- **Automake (1.17) - 1.614 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/automake/automake-1.17.tar.xz>

Soma de verificação MD5: 7ab3a02318fee6f5bd42adfc369abf10

- **Bash (5.2.32) - 10.697 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/bash/bash-5.2.32.tar.gz>

Soma de verificação MD5: f204835b2e06c06e37b5ad776ff907f4

- **Bc (6.7.6) - 463 KB:**

Página inicial: <https://git.gavinhoward.com/gavin/bc>

Transferência: <https://github.com/gavinhoward/bc/releases/download/6.7.6/bc-6.7.6.tar.xz>

Soma de verificação MD5: a47aa5e4e7395fbcd159a9228613b97b

• **Binutils (2.43.1) - 27.514 KB:**

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

Transferência: <https://sourceware.org/pub/binutils/releases/binutils-2.43.1.tar.xz>

Soma de verificação MD5: 9202d02925c30969d1917e4bad5a2320

• **Bison (3.8.2) - 2.752 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/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

• **Check (0.15.2) - 760 KB:**

Página inicial: <https://libcheck.github.io/check>

Transferência: <https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz>

Soma de verificação MD5: 50fcafcecede5a380415b12e9c574e0b2

• **Coreutils (9.5) - 5.867 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/coreutils/coreutils-9.5.tar.xz>

Soma de verificação MD5: e99adfa059a63db3503cc71f3d151e31

• **DejaGNU (1.6.3) - 608 KB:**

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

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

Soma de verificação MD5: 68c5208c58236eba447d7d6d1326b821

• **Diffutils (3.10) - 1.587 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/diffutils/diffutils-3.10.tar.xz>

Soma de verificação MD5: 2745c50f6f4e395e7b7d52f902d075bf

• **E2fsprogs (1.47.1) - 9.720 KB:**

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

Transferência: <https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.1/e2fsprogs-1.47.1.tar.gz>

Soma de verificação MD5: 75e6d1353cbe6d5728a98fb0267206cb

• **Elfutils (0.191) - 9.092 KB:**

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

Transferência: <https://sourceware.org/ftp/elfutils/0.191/elfutils-0.191.tar.bz2>

Soma de verificação MD5: 636547248fb3fae58ec48030298d3ef7

• **Expat (2.6.2) - 474 KB:**

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

Transferência: <https://prdownloads.sourceforge.net/expat/expat-2.6.2.tar.xz>

Soma de verificação MD5: 0cb75c8feb842c0794ba89666b762a2d

• **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 (5.45) - 1.218 KB:**

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

Transferência: <https://astron.com/pub/file/file-5.45.tar.gz>

Soma de verificação MD5: 26b2a96d4e3a8938827a1e572afd527a

• **Findutils (4.10.0) - 2.189 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/findutils/findutils-4.10.0.tar.xz>

Soma de verificação MD5: 870cfd71c07d37ebe56f9f4aaf4ad872

• **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 (3.9.0) - 41 KB:**

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

Transferência: https://pypi.org/packages/source/f/flit-core/flit_core-3.9.0.tar.gz

Soma de verificação MD5: 3bc52f1952b9a78361114147da63c35b

• **Gawk (5.3.0) - 3.356 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/gawk/gawk-5.3.0.tar.xz>

Soma de verificação MD5: 97c5a7d83f91a7e1b2035ebbe6ac7abd

• **GCC (14.2.0) - 90.144 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/gcc/gcc-14.2.0/gcc-14.2.0.tar.xz>

Soma de verificação MD5: 2268420ba02dc01821960e274711bde0

• **GDBM (1.24) - 1.168 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/gdbm/gdbm-1.24.tar.gz>

Soma de verificação MD5: c780815649e52317be48331c1773e987

• **Gettext (0.22.5) - 10.031 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/gettext/gettext-0.22.5.tar.xz>

Soma de verificação MD5: 3ae5580599d84be93e6213930facb2db

• **Glibc (2.40) - 18.313 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/glibc/glibc-2.40.tar.xz>

Soma de verificação MD5: b390feef233022114950317f10c4fa97



Nota

Os(As) desenvolvedores(as) da Glibc mantém uma *ramificação Git* contendo remendos considerados valiosos para a Glibc-2.40, porém infelizmente desenvolvidos depois do lançamento da Glibc-2.40. 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://ftp.gnu.org/gnu/gmp/gmp-6.3.0.tar.xz>

Soma de verificação MD5: 956dc04e864001a9c22429f761f2c283

• **Gperf (3.1) - 1.188 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz>

Soma de verificação MD5: 9e251c0a618ad0824b51117d5d9db87e

• **Grep (3.11) - 1.664 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/grep/grep-3.11.tar.xz>

Soma de verificação MD5: 7c9bbd74492131245f7cdb291fa142c0

• **Groff (1.23.0) - 7.259 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/groff/groff-1.23.0.tar.gz>

Soma de verificação MD5: 5e4f40315a22bb8a158748e7d5094c7d

• **GRUB (2.12) - 6.524 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/grub/grub-2.12.tar.xz>

Soma de verificação MD5: 60c564b1bdc39d8e43b3aab4bc0fb140

• **Gzip (1.13) - 819 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/gzip/gzip-1.13.tar.xz>

Soma de verificação MD5: d5c9fc9441288817a4a0be2da0249e29

• **Iana-Etc (20240806) - 590 KB:**

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

Transferência: <https://github.com/Mic92/iana-etc/releases/download/20240806/iana-etc-20240806.tar.gz>

Soma de verificação MD5: ea3c37c00d22f1159fc3b7d988de8476

• **Inetutils (2.5) - 1.632 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/inetutils/inetutils-2.5.tar.xz>

Soma de verificação MD5: 9e5a6dfd2d794dc056a770e8ad4a9263

• **Intltool (0.51.0) - 159 KB:**

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

Transferência: <https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz>

Soma de verificação MD5: 12e517cac2b57a0121cda351570f1e63

• **IPRoute2 (6.10.0) - 900 KB:**

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

Transferência: <https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.10.0.tar.xz>

Soma de verificação MD5: 6282e47de9c5b230e83537fba7181c9c

• **Jinja2 (3.1.4) - 235 KB:**

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

Transferência: <https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.4.tar.gz>

Soma de verificação MD5: 02ca9a6364c92e83d14b037bef4732bc

• **Kbd (2.6.4) - 1.470 KB:**

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

Transferência: <https://www.kernel.org/pub/linux/utils/kbd/kbd-2.6.4.tar.xz>

Soma de verificação MD5: e2fd7adccf6b1e98eb1ae8d5a1ce5762

• **Kmod (33) - 503 KB:**

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

Transferência: <https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-33.tar.xz>

Soma de verificação MD5: c451c4aa61521adbe8af147f498046f8

• **Less (661) - 634 KB:**

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

Transferência: <https://www.greenwoodsoftware.com/less/less-661.tar.gz>

Soma de verificação MD5: 44f54b6313c5d71fa1ac224d8d84766a

• **LFS-Bootscripts (20240825) - 34 KB:**

Transferência: <https://www.linuxfromscratch.org/lfs/downloads/12.2/lfs-bootscripts-20240825.tar.xz>

Soma de verificação MD5: 4dd8a52530eec8f8a544f833e2a82eaf

• **Libcap (2.70) - 187 KB:**

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

Transferência: <https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.70.tar.xz>

Soma de verificação MD5: df0e20c6eeca849347b87d5d6a8870c0

• **Libffi (3.4.6) - 1.360 KB:**

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

Transferência: <https://github.com/libffi/libffi/releases/download/v3.4.6/libffi-3.4.6.tar.gz>

Soma de verificação MD5: b9cac6c5997dca2b3787a59ede34e0eb

• **Libpipeline (1.5.7) - 956 KB:**

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

Transferência: <https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.7.tar.gz>

Soma de verificação MD5: 1a48b5771b9f6c790fb4efdb1ac71342

• **Libtool (2.4.7) - 996 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/libtool/libtool-2.4.7.tar.xz>

Soma de verificação MD5: 2fc0b6ddcd66a89ed6e45db28fa44232

• **Libxcrypt (4.4.36) - 610 KB:**

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

Transferência: <https://github.com/besser82/libxcrypt/releases/download/v4.4.36/libxcrypt-4.4.36.tar.xz>

Soma de verificação MD5: b84cd4104e08c975063ec6c4d0372446

• **Linux (6.10.5) - 141.739 KB:**

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

Transferência: <https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.10.5.tar.xz>

Soma de verificação MD5: 276ef1f11ed3713ec5d6f506ff55ac12



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 usuários(as) com largura de banda de velocidade limitada ou cara que desejem atualizar o kernel 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 menor.

• **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 (1.4.19) - 1.617 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz>

Soma de verificação MD5: 0d90823e1426f1da2fd872df0311298d

• **Make (4.4.1) - 2.300 KB:**

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

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

Soma de verificação MD5: c8469a3713cbb04d955d4ae4be23eeb

• **Man-DB (2.12.1) - 1.994 KB:**

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

Transferência: <https://download.savannah.gnu.org/releases/man-db/man-db-2.12.1.tar.xz>

Soma de verificação MD5: 7b044e5020aab89db41ac7ee59d6d84a

• **Man-pages (6.9.1) - 1.821 KB:**

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

Transferência: <https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.9.1.tar.xz>

Soma de verificação MD5: 4d56775b6cce4edf1e496249e7c01c1a

• **MarkupSafe (2.1.5) - 19 KB:**

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

Transferência: <https://pypi.org/packages/source/M/MarkupSafe/MarkupSafe-2.1.5.tar.gz>

Soma de verificação MD5: 8fe7227653f2fb9b1ffe7f9f2058998a

• **Meson (1.5.1) - 2.205 KB:**

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

Transferência: <https://github.com/mesonbuild/meson/releases/download/1.5.1/meson-1.5.1.tar.gz>

Soma de verificação MD5: c4f2b3e5ea632685f61ba1b833c4905c

• **MPC (1.3.1) - 756 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/mpc/mpc-1.3.1.tar.gz>

Soma de verificação MD5: 5c9bc658c9fd0f940e8e3e0f09530c62

• **MPFR (4.2.1) - 1.459 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/mpfr/mpfr-4.2.1.tar.xz>

Soma de verificação MD5: 523c50c6318dde6f9dc523bc0244690a

• **Ncurses (6.5) - 2.156 KB:**

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

Transferência: <https://invisible-mirror.net/archives/ncurses/ncurses-6.5.tar.gz>

Soma de verificação MD5: ac2d2629296f04c8537ca706b6977687

• **Ninja (1.12.1) - 235 KB:**

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

Transferência: <https://github.com/ninja-build/ninja/archive/v1.12.1/ninja-1.12.1.tar.gz>

Soma de verificação MD5: 6288992b05e593a391599692e2f7e490

• **OpenSSL (3.3.1) - 17.633 KB:**

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

Transferência: <https://www.openssl.org/source/openssl-3.3.1.tar.gz>

Soma de verificação MD5: 8a4342b399c18f870ca6186299195984

• **Patch (2.7.6) - 766 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz>

Soma de verificação MD5: 78ad9937e4caadcba1526ef1853730d5

• **Perl (5.40.0) - 13.481 KB:**

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

Transferência: <https://www.cpan.org/src/5.0/perl-5.40.0.tar.xz>

Soma de verificação MD5: cfe14ef0709b9687f9c514042e8e1e82

• **Pkgconf (2.3.0) - 309 KB:**

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

Transferência: <https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz>

Soma de verificação MD5: 833363e77b5bed0131c7bc4cc6f7747b

• **Procps (4.0.4) - 1.369 KB:**

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

Transferência: <https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.4.tar.xz>

Soma de verificação MD5: 2f747fc7df8ccf402d03e375c565cf96

• **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 (3.12.5) - 19.944 KB:**

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

Transferência: <https://www.python.org/ftp/python/3.12.5/Python-3.12.5.tar.xz>

Soma de verificação MD5: 02c7d269e077f4034963bba6befdc715

• **Documentação do Python (3.12.5) - 8.188 KB:**

Transferência: <https://www.python.org/ftp/python/doc/3.12.5/python-3.12.5-docs-html.tar.bz2>

Soma de verificação MD5: 52274d813236ca4a972fb6988480dc56

• **Readline (8.2.13) - 2.974 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/readline/readline-8.2.13.tar.gz>

Soma de verificação MD5: 05080bf3801e6874bb115cd6700b708f

• **Sed (4.9) - 1.365 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/sed/sed-4.9.tar.xz>

Soma de verificação MD5: 6aac9b2dbafcd5b7a67a8a9bcb8036c3

• **Setuptools (72.2.0) - 2.363 KB:**

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

Transferência: <https://pypi.org/packages/source/s/setuptools/setuptools-72.2.0.tar.gz>

Soma de verificação MD5: 2e0ffd0f6fc632a11442b79d9b1c68bd

- **Shadow (4.16.0) - 2.154 KB:**

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

Transferência: <https://github.com/shadow-maint/shadow/releases/download/4.16.0/shadow-4.16.0.tar.xz>

Soma de verificação MD5: eb70bad3316d08f0d3bb3d4bbeccb3b4

- **Sysklogd (2.6.1) - 452 KB:**

Página inicial: <https://www.infodrom.org/projects/sysklogd/>

Transferência: <https://github.com/troglobit/sysklogd/releases/download/v2.6.1/sysklogd-2.6.1.tar.gz>

Soma de verificação MD5: dcf0836a0fcc6568efaad230850d9c86

- **Systemd (256.4) - 15.291 KB:**

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

Transferência: <https://github.com/systemd/systemd/archive/v256.4/systemd-256.4.tar.gz>

Soma de verificação MD5: 03bd1ff158ec0bc55428c77a8f8495bd

- **Páginas de Manual do Systemd (256.4) - 676 KB:**

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

Transferência: <https://anduin.linuxfromscratch.org/LFS/systemd-man-pages-256.4.tar.xz>

Soma de verificação MD5: 8dbc0ff0d8e5e9d3565f9d2fc153310



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.

- **SysVinit (3.10) - 235 KB:**

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

Transferência: <https://github.com/slicer69/sysvinit/releases/download/3.10/sysvinit-3.10.tar.xz>

Soma de verificação MD5: b8fbe11062cf16d3b6a3709b7f6978d2

- **Tar (1.35) - 2.263 KB:**

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

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

Soma de verificação MD5: a2d8042658cfd8ea939e6d911eaf4152

- **Tcl (8.6.14) - 11.355 KB:**

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

Transferência: <https://downloads.sourceforge.net/tcl/tcl8.6.14-src.tar.gz>

Soma de verificação MD5: c30b57c6051be28fa928d09aca82841e

- **Documentação do Tcl (8.6.14) - 1.167 KB:**

Transferência: <https://downloads.sourceforge.net/tcl/tcl8.6.14-html.tar.gz>

Soma de verificação MD5: 5467198f8d57c54835bf80b98fffb0170

- **Texinfo (7.1) - 5.416 KB:**

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

Transferência: <https://ftp.gnu.org/gnu/texinfo/texinfo-7.1.tar.xz>

Soma de verificação MD5: edd9928b4a3f82674bcc3551616eef3b

- **Dados de Fuso Horário (2024a) - 444 KB:**

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

Transferência: <https://www.iana.org/time-zones/repository/releases/tzdata2024a.tar.gz>

Soma de verificação MD5: 2349edd8335245525cc082f2755d5bf4

- **Tarball do Udev-lfs (udev-lfs-20230818) - 10 KB:**

Transferência: <https://anduin.linuxfromscratch.org/LFS/udev-lfs-20230818.tar.xz>

Soma de verificação MD5: acd4360d8a5c3ef320b9db88d275dae6

- **Util-linux (2.40.2) - 8.648 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/v2.40/util-linux-2.40.2.tar.xz>

Soma de verificação MD5: 88faefc8fefced097e58142077a3d14e

- **Vim (9.1.0660) - 17.629 KB:**

Página inicial: <https://www.vim.org>

Transferência: <https://github.com/vim/vim/archive/v9.1.0660/vim-9.1.0660.tar.gz>

Soma de verificação MD5: c512a99b3704f193be1a181cc644b2b2



Nota

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

- **Wheel (0.44.0) - 99 KB:**

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

Transferência: <https://pypi.org/packages/source/w/wheel/wheel-0.44.0.tar.gz>

Soma de verificação MD5: 440ff4fe51579b7ed16f02af8f8d9494

- **XML::Parser (2.47) - 276 KB:**

Página inicial: <https://github.com/chorny/XML-Parser>

Transferência: <https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.47.tar.gz>

Soma de verificação MD5: 89a8e82cfd2ad948b349c0a69c494463

- **Xz Utils (5.6.2) - 1.277 KB:**

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

Transferência: <https://github.com//tukaani-project/xz/releases/download/v5.6.2/xz-5.6.2.tar.xz>

Soma de verificação MD5: bbf73fb28425cebb854328599f85c4cf

- **Zlib (1.3.1) - 1.478 KB:**

Página inicial: <https://zlib.net/>

Transferência: <https://zlib.net/fossils/zlib-1.3.1.tar.gz>

Soma de verificação MD5: 9855b6d802d7fe5b7bd5b196a2271655

- **Zstd (1.5.6) - 2.351 KB:**

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

Transferência: <https://github.com/facebook/zstd/releases/download/v1.5.6/zstd-1.5.6.tar.gz>

Soma de verificação MD5: 5a473726b3445d0e5d6296afd1ab6854

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/12.2/bzip2-1.0.8-install_docs-1.patch

Soma de verificação MD5: 6a5ac7e89b791aae556de0f745916f7f

- **Remendo de Correções de Internacionalização do Coreutils - 164 KB:**

Transferência: <https://www.linuxfromscratch.org/patches/lfs/12.2/coreutils-9.5-i18n-2.patch>

Soma de verificação MD5: 58961caf5bbdb02462591fa506c73b6d

• **Remendo GCC14 do Expect - 7.8 KB:**

Transferência: <https://www.linuxfromscratch.org/patches/lfs/12.2/expect-5.45.4-gcc14-1.patch>

Soma de verificação MD5: 0b8b5ac411d011263ad40b0664c669f0

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

Transferência: <https://www.linuxfromscratch.org/patches/lfs/12.2/glibc-2.40-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/12.2/kbd-2.6.4-backspace-1.patch>

Soma de verificação MD5: f75cca16a38da6caa7d52151f7136895

• **Remendo Consolidado do SysVinit - 2.5 KB:**

Transferência: <https://www.linuxfromscratch.org/patches/lfs/12.2/sysvinit-3.10-consolidated-1.patch>

Soma de verificação MD5: 17ffccb8e18c39e8cedc32046f3a475

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 a construção do sistema temporário. Nós criaremos um conjunto de diretórios em `$LFS` (no qual instalaremos as ferramentas temporárias); adicionaremos uma(m) usuária(o) desprivilegiada(o); e criaremos um ambiente apropriado de construção para aquela(e) usuária(o). Nós também explicaremos as unidades de tempo “UPCs” que usamos para medir quanto tempo leva para construir os 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) Usuário(a) LFS

Enquanto logada(o) como usuária(o) `root`, cometer um simples erro pode danificar ou destruir um sistema. Portanto, os pacotes nos próximos dois capítulos são construídos como uma(m) usuária(o) sem privilégios. Você poderia usar seu próprio nome de usuária(o), mas para tornar mais fácil configurar um ambiente de trabalho limpo, nós criaremos um(a) usuário(a) novo(a) chamado(a) `lfs` como um(a) membro(a) de um novo grupo (também chamado `lfs`) e executar comandos como `lfs` durante o processo de instalação. Como `root`, emita os seguintes comandos para adicionar a(o) novo(a) usuário(a):

```
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 o **bash** o shell padrão para o(a) usuário(a) `lfs`.

```
-g lfs
```

Essa opção acrescenta o(a) usuário(a) `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) usuário(a) nova(o).

Se quiser logar como `lfs` ou alternar para `lfs` a partir de um(a) usuário(a) não `root` (em oposição a alternar para o(a) usuário(a) `lfs` quando estiver logado(a) como `root`, o que não exige que o(a) usuário(a) `lfs` tenha uma senha, [então] você precisa configurar uma senha para `lfs`. Emita o seguinte comando como o(a) usuário(a) `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{,/*},lib,var,etc,bin,sbin,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) usuário(a) `lfs` para o segundo plano. Se o prompt "`lfs:~$`" não aparecer imediatamente, [então] emitir o comando **fg** corrigirá o problema.

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

```
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 logada(o) como usuária(o) `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 logada(o) como usuária(o) `lfs` ou quando alternado para o(a) usuário(a) `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 em execução 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
```

Configurar a máscara de criação de arquivos do(a) usuário(a) (“umask”) para “022” garante que recém criados arquivos e diretórios sejam escrevíveis somente pelos(as) donos(as) deles, mas sejam legíveis e executáveis por qualquer pessoa (assumindo que os modos padrão sejam usados pela chamada de sistema “*open(2)*”, novos arquivos terminarão com modo de permissão “644” e diretórios com modo “755”).

```
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) usuário(a) `lfs` de formas que podem afetar a construção de pacotes LFS críticos. Para assegurar que o ambiente do(a) usuário(a) `lfs` esteja limpo, verifique a presença de `/etc/bash.bashrc` e, se presente, mova-o para fora do caminho. Como o(a) usuário(a) `root`, execute:

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

Quando o(a) usuário(a) `lfs` não mais for necessária(o) (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 8.36, “Bash-5.2.32” 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 a construção das ferramentas temporárias, force o shell **bash** a ler o perfil do(a) novo(a) usuário(a):

```
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 menor 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 pode ser impreciso porque o sistema pode reagir 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 pode responder 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/12.2/> 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 isola 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 atual. 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 que contendo um conjunto de ferramentas que são conhecidas por serem boas e que estão isoladas do sistema anfitrião. Usando-se o comando **chroot**, as compilações nos capítulos subsequentes estarão isolados 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 leitores(as) novatos(as) e para prover o maior valor educacional ao mesmo tempo.

O processo de construção é baseado em *compilação cruzada*. A 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, dado que 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 de ferramentas cruzado (ou nativo). Não use os comandos no livro para um conjunto de ferramentas cruzado para algum outro propósito que não construir o LFS, a menos que você realmente entenda o que está fazendo.

Compilação cruzada envolve alguns conceitos que merecem uma seção por si próprios. Apesar que 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 aplicativos. Observe que essa máquina também é referenciada como sendo a “anfitriã”.

A anfitriã

é a máquina/sistema onde os aplicativos 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 (de vez em quando 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 aplicativos necessários para a máquina C podem ser compilados usando cc2 na rápida máquina B. Observe que a menos que B possa executar aplicativos produzidos por C, não existe maneira de testar os aplicativos recém construídos até que a própria máquina C esteja em execução. 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

Estágio	Construtora	Anfitriã	Alvo	Ação
				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 em execução. Os outros compiladores, ccA e ccC, produzem código para a máquina na qual estão em execução. 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 triplo 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 “trio” 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 trio 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 inequívoca. 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 “trio” permanece embutida no léxico. Uma maneira simples para determinar o seu trio do sistema é a de executar o script `config.guess` que vem com o fonte para muitos pacotes. Desempacote os fontes do `binutils`, execute o script `./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 maior parte 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 aplicativo, prepara o aplicativo 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 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 oficial cobrindo todas as plataformas está em [uma página wiki da Glibc](#).

Para a finalidade de falsificar uma compilação cruzada no LFS, o nome do trio do anfitrião é ligeiramente ajustado mudando-se o campo `"vendor"` na variável `LFS_TGT`, de forma que diga `"lfs"`. Nós também usamos a opção `--with-sysroot` quando da construção do vinculador cruzado e do compilador cruzado para informá-los onde encontrar

os arquivos necessários do anfitrião. Isso assegura que nenhum dos outros aplicativos construídos no Capítulo 6 consegue se vincular a bibliotecas na máquina de construção. Somente dois estágios são obrigatórios e mais um para testes.

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	Reconstruir e testar cc-lfs usando cc-lfs em 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 apenas um compilador; 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 precisa ser vinculada à biblioteca glibc para ser completamente funcional. Além disso, a biblioteca padrão para C++ (libstdc++) também precisa estar vinculada com a glibc. A solução para esse problema de ovo e galinha é a de primeiro construir uma libgcc degradada baseada em cc1, faltando algumas funcionalidades, tais como camadas e manuseio de exceções, e então construir a glibc usando esse compilador degradado (a própria glibc não é degradada), e também construir a libstdc++. Essa última biblioteca carecerá de algumas das funcionalidades da libgcc.

O resultado do parágrafo precedente é o de que cc1 é inapto para construir uma libstdc++ completamente funcional com a libgcc degradada, porém cc1 é o único compilador disponível para construir as bibliotecas C/C++ durante o estágio 2. Existem duas razões pelas quais nós não usamos imediatamente o compilador construído no estágio 2, cc-lfs, para construir essas bibliotecas.

- Falando genericamente, cc-lfs não consegue executar em pc (o sistema anfitrião). Ainda que os trios para pc e lfs sejam compatíveis entre si, um executável para lfs precisa depender da glibc-2.40; a distribuição anfitriã possivelmente utilize ou uma implementação diferente da libc (por exemplo, musl), ou um lançamento anterior da glibc (por exemplo, glibc-2.13).
- Ainda se cc-lfs conseguisse executar em pc, usá-la em pc criaria um risco de vinculação às bibliotecas de pc, dado que cc-lfs é um compilador nativo.

Assim, quando nós construirmos gcc estágio 2, nós instruímos o sistema de construção a reconstruir libgcc e libstdc++ com cc1, porém nós vinculamos libstdc++ à libgcc reconstruída recentemente, em vez da antiga, degradada construção. Isso torna a libstdc++ reconstruída completamente funcional.

No Capítulo 8 (ou “estágio 3”), todos os pacotes necessários para o sistema LFS são construídos. Ainda se um pacote já tenha sido instalado no sistema LFS em um capítulo anterior, nós ainda reconstruímos o pacote. A razão principal para reconstruir esses pacotes é a de torná-los estáveis: se nós reinstalarmos um pacote LFS em um sistema LFS completo, [então] o conteúdo reinstalado do pacote deveria ser o mesmo que o conteúdo do mesmo pacote quando primeiro instalado no Capítulo 8. Os pacotes temporários instalados no Capítulo 6 ou no Capítulo 7 não conseguem satisfazer essa exigência, pois alguns deles são construídos sem dependências opcionais e o autoconf não consegue realizar algumas verificações de recursos no Capítulo 6, por causa da compilação cruzada, causando nos pacotes temporários a falta de recursos opcionais ou o uso de rotinas sub ótimas de código. Adicionalmente, uma razão menor para reconstruir os pacotes é a de executar as suítes de teste.

Outros Detalhes Procedurais

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

O Binutils é instalado primeiro, pois as execuções do **configure** de ambos gcc e glibc realizam vários testes de recursos no montador e no vinculador 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.

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

O próximo pacote instalado é o 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 script de configuração do gcc não procura nos diretórios do PATH para encontrar quais ferramentas usar. Entretanto, durante a operação atual do próprio gcc, os mesmos caminhos de procura não são necessariamente usados. Para descobrir qual vinculador 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 do gcc passando-se a opção de linha de comando `-v` enquanto compilar um aplicativo. Por exemplo, **`$LFS_TGT-gcc -v example.c`** (ou sem **`$LFS_TGT-`** se retornar posteriormente) exibirá informação detalhada acerca do preprocessador, 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 a glibc. As considerações mais importantes para construir glibc são o compilador, as ferramentas binárias e os cabeçalhos do núcleo. O compilador e as ferramentas binárias geralmente não são um problema, pois a glibc sempre passará aqueles relacionados ao parâmetro `--host` para o script de configuração dela; por

exemplo, em nosso caso, o compilador será `$LFS_TGT-gcc` e a ferramenta `readelf` será `$LFS_TGT-readelf`. Os cabeçalhos do núcleo podem ser um pouco mais complicados. Portanto, nós não assumimos riscos e usamos a chave de configuração disponível para impor a seleção correta. Depois da execução do `configure`, verifique o conteúdo do arquivo `config.make` no diretório `build` para todos os detalhes importantes. Esses itens destacam um aspecto importante do pacote `glibc`—ele é muito autossuficiente em termos do maquinário dele de construção e geralmente não depende de padrões do conjunto de ferramentas.

Como mencionado acima, a biblioteca C++ padrão é compilada depois, seguida no Capítulo 6 por outros aplicativos que precisam ser compilados cruzadamente para quebrar dependências circulares em tempo de construção. A etapa de instalação de todos aqueles pacotes usa a variável `DESTDIR` para forçar a instalação no sistema de arquivos do LFS.

Ao final do Capítulo 6 o compilador nativo do LFS é instalado. Primeiro `binutils` passagem 2 é construído, no mesmo diretório `DESTDIR` que os outros aplicativos, então a segunda passagem do `gcc` é construída, omitindo algumas bibliotecas não críticas. Devido a algumas lógicas estranhas no script `configure` do `gcc`, `CC_FOR_TARGET` termina como `cc` quando o anfitrião for o mesmo que o alvo, porém for diferente do sistema de construção. Essa é a razão pela qual `CC_FOR_TARGET=$LFS_TGT-gcc` é declarado explicitamente como uma das opções de configuração.

Uma vez dentro do ambiente `chroot` no Capítulo 7, as instalações temporárias de aplicativos 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 a partir de pacotes atualizados. Misturar as instruções de diferentes versões do livro LFS pode causar quebras sutis. Esse tipo de problema geralmente é resultado da reutilização de algum script criado para um lançamento anterior do LFS. Tal reutilização é fortemente desencorajada. Se você estiver reutilizando scripts para um lançamento anterior do LFS por qualquer motivo, [então] você precisará ter muito cuidado para atualizar os scripts 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 é 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 quando da aplicação de um remendo. Não se preocupe com esses alertas; o remendo ainda foi aplicado com sucesso.
- Durante a compilação da maior parte 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 script pequeno 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 aplicativo **tar**, extraia o pacote para ser construído. Em Capítulo 5 e Capítulo 6, certifique-se de que você seja o(a) usuário(a) *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 de Ferramentas Cruzado

5.1. Introdução

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

Os aplicativos 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, dado que elas pertencem ao sistema que queremos construir.

5.2. Binutils-2.43.1 - Passagem 1

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

Tempo aproximado de construção: 1 UPC

Espaço em disco exigido: 677 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:

`--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 8.20.2, “Conteúdo do Binutils.”

5.3. GCC-14.2.0 - Passagem 1

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

Tempo aproximado de construção: 3,2 UPC

Espaço em disco exigido: 4,9 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 deste capítulo. Os procedimentos são os mesmos que todos os outros capítulos, conforme explicados anteriormente (Instruções de construção de pacote). Primeiro, extraia o tarball gcc-14.2.0 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-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 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
```

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=2.40 \
  --with-sysroot=$LFS \
  --with-newlib \
  --without-headers \
  --enable-default-pie \
  --enable-default-ssp \
  --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=2.40

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-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:



Nota

O comando abaixo mostra um exemplo da substituição de comando aninhada usando dois métodos: aspas invertidas e uma construção `$()`. Poderia ser reescrito usando o mesmo método para ambas as substituições, porém é mostrado dessa maneira para demonstrar o como eles podem ser misturados. Geralmente o método `$()` é preferido.

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

Detalhes acerca deste pacote estão localizados na Seção 8.29.2, “Conteúdo do GCC.”

5.4. Cabeçalhos da API do Linux-6.10.5

Os Cabeçalhos da API do Linux (em linux-6.10.5.tar.xz) expõem a API do kernel para uso pela Glibc.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 1,6 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 kernel visíveis para o(a) usuário(a) 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-2.40

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 1,3 UPC

construção:

Espaço em disco exigido: 828 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-2.40-fhs-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) \
  --enable-kernel=4.19 \
  --with-headers=$LFS/usr/include \
  --disable-nscd \
  libc_cv_slibdir=/usr/lib
```

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=4.19
```

Isso diz para a Glibc para compilar a biblioteca com suporte para núcleos Linux 4.19 e posteriores. Contornos para núcleos mais antigos não estão habilitados.

```
--with-headers=$LFS/usr/include
```

Isso diz para a Glibc para compilar a si mesma contra os cabeçalhos recentemente instalados no diretório `$LFS/usr/include`, de forma que ela saiba exatamente quais recursos o núcleo tem e possa otimizar-se adequadamente.

```
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 quando construir-se como um “make paralelo”. Se isso ocorrer, [então] 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` de `make` é usada por quase todos os pacotes para definir o local onde o pacote deveria ser instalado. Se ela não estiver configurada, [então] o padrão é 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
```



Cuidado

Neste ponto, é imperativo parar e certificar-se de que as funções básicas (compilar e lincar) do novo conjunto de ferramentas estão funcionando como esperado. Para realizar uma verificação de sanidade, execute os seguintes comandos:

```
echo 'int main(){}' | $LFS_TGT-gcc -xc -
readelf -l a.out | grep ld-linux
```

Se tudo estiver funcionando corretamente, [então] não deveriam existir quaisquer erros e a saída do comando final será na forma:

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

Observe que, para máquinas de 32 bits, o nome do interpretador será `/lib/ld-linux.so.2`.

Se a saída gerada não for mostrada como acima ou não existir saída gerada nenhuma, então alguma coisa está errada. Investigue e refaça os passos para descobrir onde está o problema e corrija-o. Esse problema precisa ser resolvido antes de continuar.

Uma vez que tudo esteja bem, limpe o arquivo de teste:

```
rm -v a.out
```



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-14.2.0

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 0,2 UPC

construção:

Espaço em disco exigido: 1,2 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-14.2.0`.

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) \
--prefix=/usr \
--disable-multilib \
--disable-nls \
--disable-libstdcxx-pch \
--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/14.2.0
```

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`.

`--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++/14.2.0`

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++/14.2.0`. 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 8.29.2, “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 lembrar 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 usuário(a) `lfs`, com o ambiente conforme descrito na Seção 4.4, “Configurando o Ambiente.”

6.2. M4-1.4.19

O pacote M4 contém um processador de macro.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 31 MB

6.2.1. Instalação do M4

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 8.13.2, “Conteúdo de M4.”

6.3. Ncurses-6.5

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

Tempo aproximado de 0,4 UPC

construção:

Espaço em disco exigido: 53 MB

6.3.1. Instalação do Ncurses

Primeiro, assegure que **gawk** é encontrado primeiro durante a configuração:

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

Então, execute os seguintes comandos para construir o aplicativo “tic” no anfitrião de construção:

```
mkdir build
pushd build
./configure
make -C include
make -C progs tic
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
```

O significado das novas opções de configuração:

--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 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.

--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.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS TIC_PATH=$(pwd)/build/progs/tic 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 do install:

```
TIC_PATH=$(pwd)/build/progs/tic
```

Nós precisamos passar o caminho do recém construído aplicativo **tic** que executa na máquina de construção, de forma que a base de dados de terminal possa ser criada sem erros.

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 8.30.2, "Conteúdo do Ncurses."

6.4. Bash-5.2.32

O pacote Bash contém o Bourne-Again SHell.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 67 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 \
            bash_cv_strtold_broken=no
```

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 8.36.2, “Conteúdo do Bash.”

6.5. Coreutils-9.5

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

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 175 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 \
            --enable-no-install-program=kill,uptime
```

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"/'
```

Detalhes acerca deste pacote estão localizados na Seção 8.58.2, “Conteúdo do Coreutils.”

6.6. Diffutils-3.10

O pacote Diffutils contém aplicativos que mostram as diferenças entre arquivos ou diretórios.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 29 MB

6.6.1. Instalação do Diffutils

Prepare o Diffutils 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 8.60.2, “Conteúdo do Diffutils.”

6.7. File-5.45

O pacote File contém um utilitário para determinar o tipo de um dado arquivo ou arquivos.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 37 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.la
```

Detalhes acerca deste pacote estão localizados na Seção 8.11.2, “Conteúdo do File.”

6.8. Findutils-4.10.0

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: 48 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 8.62.2, “Conteúdo do Findutils.”

6.9. Gawk-5.3.0

O pacote Gawk contém aplicativos para manipular arquivos de texto.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 47 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 8.61.2, “Conteúdo do Gawk.”

6.10. Grep-3.11

O pacote Grep contém aplicativos para procura ao longo do conteúdo de arquivos.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 27 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 8.35.2, “Conteúdo do Grep.”

6.11. Gzip-1.13

O pacote Gzip contém aplicativos para comprimir e descomprimir arquivos.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 11 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 8.65.2, “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 \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

O significado da nova opção do configure:

--without-guile

Apesar de nós estarmos compilando cruzadamente, o configure tenta usar o guile oriundo do anfitrião de construção se encontrá-lo. Isso provoca falha de compilação, de forma que essa chave evita usá-lo.

Compile o pacote:

```
make
```

Instale o pacote:

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

Detalhes acerca deste pacote estão localizados na Seção 8.69.2, “Conteúdo do Make.”

6.13. Patch-2.7.6

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: 12 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 8.70.2, “Conteúdo do Patch.”

6.14. Sed-4.9

O pacote Sed contém um editor de fluxo.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 21 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 8.31.2, “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 0,1 UPC

construção:

Espaço em disco exigido: 42 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 8.71.2, “Conteúdo do Tar.”

6.16. Xz-5.6.2

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: 20 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-5.6.2
```

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-2.43.1 - Passagem 2

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

Tempo aproximado de 0,4 UPC

construção:

Espaço em disco exigido: 549 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 '6009s/$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 configuração:

--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 8.20.2, “Conteúdo do Binutils.”

6.18. GCC-14.2.0 - Passagem 2

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

Tempo aproximado de 4,2 UPC

construção:

Espaço em disco exigido: 5,5 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-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

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
```

Substitua a regra de construção dos cabeçalhos da libgcc e da libstdc++ para permitir construir essas bibliotecas com suporte a camadas POSIX:

```
sed '/thread_header =/s/@.*@/gthr-posix.h/' \
-i libgcc/Makefile.in libstdc++-v3/include/Makefile.in
```

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=$LFS_TGT \
LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc \
--prefix=/usr \
--with-build-sysroot=$LFS \
--enable-default-pie \
--enable-default-ssp \
--disable-nls \
--disable-multilib \
--disable-libatomic \
--disable-libgomp \
--disable-libquadmath \
--disable-lsanitizer \
--disable-libssp \
--disable-libvtv \
--enable-languages=c,c++
```

O significado das novas opções de configuração:

```
--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 `$LFS`. Porém, o sistema de construção para o GCC usa outras ferramentas, que não estão cientes desse local. Essa chave é necessária, de forma que tais ferramentas procurarão pelos arquivos necessários em `$LFS`, e não no anfitrião.

```
--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.

```
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`.

```
--disable-libsanitizer
```

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.

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 8.29.2, “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 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 usuário(a) `root` e não mais como usuário(a) `lfs`. Também, verifique duplamente se `$LFS` está configurada no ambiente do(a) `root`.

Atualmente, a hierarquia de diretório inteira em `$LFS` é de propriedade do(a) usuário(a) `lfs`, um(a) usuário(a) que existe somente no sistema anfitrião. Se os diretórios e os arquivos sob `$LFS` forem mantidos como estão, [então] eles serão de propriedade de um ID de usuária(o) sem uma conta correspondente. Isso é perigoso, pois uma conta de usuária(o) criada posteriormente poderia obter esse mesmo ID de usuária(o) e se tornaria proprietária(o) de todos os arquivos sob `$LFS`, dessa forma expondo esses arquivos a possível manipulação maliciosa.

Para endereçar esse problema, mude a propriedade dos diretórios `$LFS/*` para usuária(o) `root` executando o seguinte comando:

```
chown --from lfs -R root:root $LFS/{usr,lib,var,etc,bin,sbin,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

Os aplicativos executando no espaço do(a) usuário(a) 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 daemon `udev` possivelmente mude a propriedade ou as permissões dos nós de dispositivo criados pelo núcleo e crie novos nós de dispositivo ou links 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`, [então] 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 de 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` são titularizados pelo ID de grupo 5. Esse é o ID que 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(a), escrevível por grupo). Juntamente com a opção acima, isso garante que o `devpts` criará nós de dispositivos que atendam às exigências de `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 link 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 usuário(o) `root`, execute o seguinte comando para entrar no ambiente que é, neste momento, povoado apenas com as 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 `home` do(a) usuário(a) `root` e outra para os diretórios para arquivos temporários.

A primeira mudança de modo assegura que nem toda pessoa possa entrar no diretório `/root`—o mesmo que um(a) usuário(a) normal faria com o próprio diretório `home` dele ou dela. A segunda mudança de modo garante que qualquer usuário(a) consiga escrever nos diretórios `/tmp` e `/var/tmp`, mas não consiga remover deles os arquivos de outros(as) usuários(as). 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, sinta-se 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 Links Simbólicos

Historicamente, o Linux manteve uma lista dos sistemas de arquivos montados no arquivo `/etc/mtab`. Os Núcleos modernos mantêm essa lista internamente e a expõem para o(a) usuário(a) via sistema de arquivos `/proc`. Para satisfazer utilitários que esperam encontrar o `/etc/mtab`, crie o seguinte link 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) usuário(a) `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
uidd:x:80:80:UID Generation Daemon User:/dev/null:/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:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uidd:x:80:
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 usuárias(os) ou grupos específicas(os). A Linux Standard Base (LSB, disponível em <http://refspecs.linuxfoundation.org/lsb.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 `/etc/fstab` 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 aplicativos 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 usuário(a) separados para usuárias(as) e grupos não mapeados(as) (aqueles(as) existem no servidor NFS ou no espaço de nome de usuário pai, porém “não existem” na máquina local ou no espaço de nome separado). Nós atribuímos `nobody` e `nogroup` para evitar um ID não nomeado. Porém, outras distribuições possivelmente tratem esse ID diferentemente, de forma que qualquer aplicativo portátil não deveria depender dessa atribuição.

Alguns pacotes precisam de um código de idioma.

```
localedef -i C -f UTF-8 C.UTF-8
```

Alguns testes no Capítulo 8 precisam de um(a) usuário(a) regular. Nós adicionamos esse(a) usuário(a) 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 usuária(o) e 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 usuária(o) se logou pela última vez. O arquivo `/var/log/faillog` registra tentativas de login falhas. O arquivo `/var/log/btmp` registra tentativas de login inválidas.



Nota

O arquivo `/run/utmp` registra as(os) usuárias(os) que estão atualmente logadas(os). Esse arquivo é criado dinamicamente nos scripts de inicialização.



Nota

Os arquivos `utmp`, `wtmp`, `btmp` e `lastlog` usam números inteiros de 32 bits para carimbo de tempo e eles serão fundamentalmente quebrados depois do ano 2038. Muitos pacotes pararam de usá-los e outros pacotes pararão de usá-los. Não confie no conteúdo deles para nada. Provavelmente é melhor considerá-los obsoletos.

7.7. Gettext-0.22.5

O pacote Gettext contém utilitários para internacionalização e localização. Eles permitem que aplicativos sejam compilados com Suporte ao Idioma Nativo (Native Language Support - NLS), habilitando-os a emitir mensagens no idioma nativo do(a) usuário(a).

Tempo aproximado de construção: 1,1 UPC

Espaço em disco exigido: 321 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 8.33.2, “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: 57 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 8.34.2, “Conteúdo do Bison.”

7.9. Perl-5.40.0

O pacote Perl contém o Practical Extraction and Report Language.

Tempo aproximado de construção: 0,6 UPC

Espaço em disco exigido: 285 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/perl5/5.40/core_perl \
-D archlib=/usr/lib/perl5/5.40/core_perl \
-D sitelib=/usr/lib/perl5/5.40/site_perl \
-D sitearch=/usr/lib/perl5/5.40/site_perl \
-D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.40/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" (5.40) 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 5.40.0) sem reinstalar todos os módulos.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.43.2, "Conteúdo do Perl."

7.10. Python-3.12.5

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: 0,4 UPC

Espaço em disco exigido: 603 MB

7.10.1. 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-3.12.5.tar.xz` (perceba a primeira letra maiúscula).

Prepare o Python para compilação:

```
./configure --prefix=/usr \
            --enable-shared \
            --without-ensurepip
```

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.

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 8.52.2, “Conteúdo do Python 3.”

7.11. Texinfo-7.1

O pacote Texinfo contém aplicativos para leitura, escrita e conversão de páginas info.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 130 MB

7.11.1. 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 8.72.2, “Conteúdo do Texinfo.”

7.12. Util-linux-2.40.2

O pacote Util-linux contém diversos aplicativos utilitários.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 180 MB

7.12.1. 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-2.40.2
```

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 **uuid** e `libuuid`.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados na Seção 8.79.2, “Conteúdo do Util-linux.”

7.13. Limpando e Salvando o Sistema Temporário

7.13.1. 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
```

7.13.2. 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 seu sistema anfitrião. Tome cuidado extra acerca dos comandos que você vai executar, uma vez que erros cometidos aqui podem modificar seu sistema anfitrião. Esteja ciente de que a variável de ambiente `LFS` está configurada para usuário(a) `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`.”

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 home do(a) usuário(a) `root` do sistema anfitrião, o qual tipicamente é encontrado no sistema de arquivos raiz. Substitua `$HOME` por um diretório da sua escolha se não quiser ter a cópia de segurança armazenada no diretório home 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-12.2.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.

7.13.3. Restauro

No caso de alguns erros tiverem 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) usuário(a) `root` e você não mudar para o diretório `$LFS` ou a variável de ambiente `LFS` não estiver configurada para o(a) usuário(a) `root`, [então] isso destruirá seu sistema anfitrião inteiro. **VOCÊ ESTÁ AVISADO(A).**

```
cd $LFS
rm -rf /*
tar -xpf $HOME/lfs-temp-tools-12.2.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, [então] lembre-se de verificar se os sistemas de arquivos virtuais ainda estão montados (`findmnt | grep $LFS`). Se eles não estiverem montados, [então] 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 forma 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 ao 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) usuário(a) 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 essas técnicas ou possivelmente seja uma combinação de duas ou mais dessas técnicas. Esta seção menciona brevemente problemas que possivelmente surjam quando da atualização de 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), [então] nada mais precisa ser reconstruído. O sistema seguirá funcionando bem graças à interface bem definida entre o núcleo e o espaço de usuária(o). Especificamente, os cabeçalhos da API do Linux não precisam ser atualizados juntamente com o núcleo. Você meramente precisará reiniciar 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-2.40), [então] algumas etapas extras serão necessárias para evitar quebrar o sistema. Leia-se Seção 8.5, “Glibc-2.40” para detalhes.
- Se um pacote contendo uma biblioteca compartilhada for atualizado e se o nome da biblioteca mudar, então quaisquer pacotes dinamicamente vinculados à biblioteca precisam ser recompilados, para vincular contra a biblioteca mais nova. (Observe que não existe correlação entre a versão de pacote e o nome da biblioteca). 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` com 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) vinculado aos nomes antigo e novo de uma biblioteca compartilhada (por exemplo, o pacote aponta para `libfoo.so.2` e `libbar.so.1`, enquanto o último se vincula a `libfoo.so.3`), [então] 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 vinculados à antiga biblioteca compartilhada depois que o pacote que fornece a biblioteca compartilhada for atualizado. Para evitar o problema, os(as) usuários(as) precisarão reconstruir cada pacote vinculado 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 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) usuários(as) 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 de gerenciamento de pacote anterior. Cada pacote é instalado como no esquema anterior. Mas, em vez de fazer o link simbólico via um nome genérico de pacote, cada arquivo é simbolicamente vinculado à hierarquia `/usr`. Isso remove a necessidade de expandir as variáveis de ambiente. Ainda que os links simbólicos possam ser criados pelo(a) usuário(a), muitos gerenciadores de pacote usam essa abordagem e automatizam a criação dos links 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 Usuária(o)

Esse esquema, único para LFS, foi concebido por Matthias Benkmann e está disponível a partir do *Hints Project*. Nesse esquema, cada pacote é instalado como uma(m) usuária(o) separada(o) nos locais padrão. Arquivos pertencentes a um pacote são facilmente identificados checando o ID de usuária(o). As características e deficiências dessa abordagem são muito complexas para serem descritas 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`, `/etc/ld.so.conf`, `/etc/sysconfig/rc.site`, `/etc/sysconfig/network` e `/etc/sysconfig/ifconfig.eth0`.

Um núcleo personalizado possivelmente seja necessário para o novo sistema, dependendo das diferenças no hardware do sistema e a configuração original do núcleo.



Nota

Tem havido alguns relatos de problemas quando da cópia entre arquiteturas similares, porém não idênticas. Por exemplo, o conjunto de instruções para um sistema Intel não é idêntico às instruções do processador AMD, e versões posteriores de alguns processadores possivelmente forneçam instruções que estão indisponíveis em versões anteriores.

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-6.9.1

O pacote Man-pages contém mais que 2.400 páginas de manual.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 52 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 prefix=/usr install
```

8.3.2. Conteúdo das Páginas de Manual

Arquivos instalados: várias páginas de manual

Descrições Curtas

<code>man pages</code>	Descreve funções da linguagem de programação C, arquivos importantes de dispositivo e arquivos significantes de configuração
------------------------	--

8.4. Iana-Etc-20240806

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 services protocols /etc
```

8.4.2. Conteúdo do Iana-Etc

Arquivos instalados: /etc/protocols e /etc/services

Descrições Curtas

<code>/etc/protocols</code>	Descreve os vários protocolos DARPA da Internet que estão disponíveis a partir do subsistema TCP/IP
<code>/etc/services</code>	Fornecer 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-2.40

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 12 UPC

construção:

Espaço em disco exigido: 3,1 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-2.40-fhs-1.patch
```

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

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

Garanta que os utilitários **ldconfig** e **sln** serão instalados no `/usr/sbin`:

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

Prepare a Glibc para compilação:

```
../configure --prefix=/usr          \
              --disable-werror      \
              --enable-kernel=4.19  \
              --enable-stack-protector=strong \
              --disable-nscd        \
              libc_cv_slibdir=/usr/lib
```

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=4.19

Essa opção diz ao sistema de construção que esta Glibc possivelmente seja usada com núcleos tão antigos quanto 4.19. 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. Umhas poucas falhas saídas de mais que 5.000 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.

- Além disso, 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*).

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-2.40) 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 8.27, “Libxcrypt-4.4.36.” Além de uma instalação normal da Libxcrypt, **você DEVE seguir a observação na seção Libxcrypt para instalar a libxcrypt.so.1* (substituindo libxcrypt.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
```

- Atualize o núcleo e reinicialize se ele for mais antigo que 4.19 (verifique a versão atual com **uname -r**) ou se quiser atualizá-lo mesmo assim, seguindo a Seção 10.3, “Linux-6.10.5.”
- Atualize os cabeçalhos da API do núcleo se forem mais antigos que 4.19 (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-6.10.5” (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.

Instale o pacote:

```
make install
```

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

```
sed '/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 SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
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-2.40/localedata/SUPPORTED` (inclui cada localidade listada acima e muitas mais) de uma vez com o seguinte comando consumidor de tempo:

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

Então, use o comando **localedef** para criar e instalar localidades não listadas no arquivo `glibc-2.40/localedata/SUPPORTED` quando você precisar delas. Por exemplo, as seguintes duas localidades são necessárias para alguns testes posteriormente neste capítulo:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```



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
group: files
shadow: files

hosts: files 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 ../../tzdata2024a.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
```

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 são 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

Aplicativos instalados:	<code>gencat</code> , <code>getconf</code> , <code>getent</code> , <code>iconv</code> , <code>iconvconfig</code> , <code>ldconfig</code> , <code>ldd</code> , <code>lddlibc4</code> , <code>ld.so</code> (link simbólico para <code>ld-linux-x86-64.so.2</code> ou <code>ld-linux.so.2</code>), <code>locale</code> , <code>localedef</code> , <code>makedb</code> , <code>mtrace</code> , <code>pcprofiledump</code> , <code>pldd</code> , <code>sln</code> , <code>sotruss</code> , <code>sprof</code> , <code>tzselect</code> , <code>xtrace</code> , <code>zdump</code> e <code>zic</code>
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 ao(à) usuário(a) a respeito do local do sistema e relata 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
ld-*.so	O aplicativo auxiliar para executáveis de bibliotecas compartilhadas
libBrokenLocale	Usado internamente pela Glibc como um hack grosseiro para executar aplicativos quebrados (por exemplo, alguns aplicativos Motif). Vejam-se comentários em <code>glibc-2.40/locale/broken_cur_max.c</code> para mais informação
libanl	Biblioteca fictícia que não contém funções. Anteriormente era a biblioteca assíncrona de pesquisa de nome, cujas funções agora estão em <code>libc</code>
libc	A biblioteca principal C
libc_malloc_debug	Liga verificação de alocação de memória quando pré-carregada
libdl	Biblioteca fictícia que não contém funções. Anteriormente era a biblioteca de interface do vinculador dinâmico, cujas funções agora estão em <code>libc</code>
libg	Biblioteca fictícia que não contém funções. Anteriormente era uma biblioteca de tempo de execução para g++
libm	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 usuárias(os), nomes de grupos, pseudônimos, serviços, protocolos, etc. 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-1.3.1

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: 6,4 MB

8.6.1. Instalação do Zlib

Prepare o 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 menos que 0,1 UPC

construção:

Espaço em disco exigido: 7,2 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 -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

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

Aplicativos instalados: bunzip2 (link para bzip2), bzcat (link para bzip2), bzcmp (link para bzdiff), bzdiff, 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
bzdiff	Executa diff 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
bzmore	Executa more em arquivos compactados com bzip
<code>libbz2</code>	A biblioteca que implementa compressão de dados de classificação de blocos sem perdas, usando o algoritmo Burrows-Wheeler

8.8. Xz-5.6.2

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 porcentagem de compressão que com os tradicionais comandos **gzip** ou **bzip2**.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 21 MB

8.8.1. Instalação do Xz

Prepare Xz para compilação com:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.6.2
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.8.2. Conteúdo do Xz

Aplicativos instalados: [lzcat](#) (link para xz), [lzcmp](#) (link para xzdiff), [lzdifff](#) (link para xzdiff), [lzegrep](#) (link para xzgrep), [lzfgrep](#) (link para xzgrep), [lzugrep](#) (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-5.6.2

Descrições Curtas

lzcat	Descomprime para a saída padrão
lzcmp	Executa cmp em arquivos comprimidos LZMA
lzdifff	Executa diff em arquivos comprimidos LZMA
lzegrep	Executa egrep em arquivos comprimidos LZMA
lzfgrep	Executa fgrep em arquivos comprimidos LZMA
lzugrep	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
<code>liblzma</code>	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

Aplicativos 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-1.5.6

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: 84 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

Aplicativos 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-5.45

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: 17 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

Aplicativos 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-8.2.13

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: 16 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
```

Prepare Readline para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --with-curses \
            --docdir=/usr/share/doc/readline-8.2.13
```

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 contra a biblioteca `libncursesw`.

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make SHLIB_LIBS="-lncursesw" install
```

Se desejado, instale a documentação:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.2.13
```

8.12.2. Conteúdo do Readline

Bibliotecas instaladas: libhistory.so e libreadline.so

Diretórios instalados: /usr/include/readline e /usr/share/doc/readline-8.2.13

Descrições Curtas

`libhistory` Fornece uma consistente interface de usuária(o) para re-chamar linhas do histórico

`libreadline`

Fornece um conjunto de comandos para manipular texto digitado em uma sessão interativa de um aplicativo

8.13. M4-1.4.19

O pacote M4 contém um processador de macro.

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 48 MB

8.13.1. 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
```

8.13.2. 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) usuário(a) 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, etc. O aplicativo **m4** pode ser usado ou como um front-end para um compilador ou como um processador de macro independente

8.14. Bc-6.7.6

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

8.14.1. Instalação do Bc

Prepare Bc para compilação:

```
CC=gcc ./configure --prefix=/usr -G -O3 -r
```

O significado das opções do configure:

CC=gcc

Esse parâmetro especifica o compilador a 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
```

8.14.2. Conteúdo do Bc

Aplicativos instalados: bc e dc

Descrições Curtas

bc Uma calculadora de linha de comando

dc Uma calculadora de linha de comando de polonesa - reversa

8.15. Flex-2.6.4

O pacote Flex contém um utilitário para gerar aplicativos que reconhecem padrões em texto.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 33 MB

8.15.1. Instalação do Flex

Prepare Flex para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Compile o pacote:

```
make
```

Para testar os resultados (cerca de 0,5 UPC), 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
```

8.15.2. Conteúdo do Flex

Aplicativos 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**
- `libfl` A biblioteca `flex`

8.16. Tcl-8.6.14

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: 3,2 UPC

Espaço em disco exigido: 91 MB

8.16.1. 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|${SRCDIR}/unix/pkgs/tdbc1.1.7|/usr/lib/tdbc1.1.7|" \
    -e "s|${SRCDIR}/pkgs/tdbc1.1.7/generic|/usr/include|" \
    -e "s|${SRCDIR}/pkgs/tdbc1.1.7/library|/usr/lib/tcl8.6|" \
    -e "s|${SRCDIR}/pkgs/tdbc1.1.7|/usr/include|" \
    -i pkgs/tdbc1.1.7/tdbcConfig.sh

sed -e "s|${SRCDIR}/unix/pkgs/itcl4.2.4|/usr/lib/itcl4.2.4|" \
    -e "s|${SRCDIR}/pkgs/itcl4.2.4/generic|/usr/include|" \
    -e "s|${SRCDIR}/pkgs/itcl4.2.4|/usr/include|" \
    -i pkgs/itcl4.2.4/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:

```
make test
```

Instale o pacote:

```
make install
```

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 /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

Opcionalmente, instale a documentação emitindo os seguintes comandos:

```
cd ..
tar -xf ../tcl8.6.14-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.14
cp -v -r ./html/* /usr/share/doc/tcl-8.6.14
```

8.16.2. Conteúdo do Tcl

Aplicativos 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

8.17. 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

8.17.1. 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 catastróficamente e outras falhas sutis possivelmente também aconteçam.

Agora, faça algumas mudanças para permitir o pacote com `gcc-14.1` ou posterior:

```
patch -Np1 -i ../expect-5.45.4-gcc14-1.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
```

8.17.2. 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)

8.18. 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 menos que 0,1 UPC

construção:

Espaço em disco exigido: 6,9 MB

8.18.1. 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/dejagnum.html ../doc/dejagnum.texi
makeinfo --plaintext      -o doc/dejagnum.txt  ../doc/dejagnum.texi
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
install -v -dm755 /usr/share/doc/dejagnum-1.6.3
install -v -m644 doc/dejagnum.{html,txt} /usr/share/doc/dejagnum-1.6.3
```

8.18.2. Conteúdo do DejaGNU

Aplicativo instalado: dejagnum e runtest

Descrições Curtas

dejagnum Iniciador de comando auxiliar DejaGNU

runtest Um script encapsulador que localiza o shell **expect** adequado e, em seguida, executa o DejaGNU

8.19. Pkgconf-2.3.0

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: 4,7 MB

8.19.1. Instalação do Pkgconf

Prepare Pkgconf para compilação:

```
./configure --prefix=/usr          \
            --disable-static       \
            --docdir=/usr/share/doc/pkgconf-2.3.0
```

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

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
```

8.19.2. Conteúdo do Pkgconf

Aplicativos instalados: pkgconf, pkg-config (link para pkgconf) e bomtool
Biblioteca instalada: libpkgconf.so
Diretório instalado: /usr/share/doc/pkgconf-2.3.0

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"
<code>libpkgconf</code>	Contém a maior parte da funcionalidade do "pkgconf", enquanto permite que outras ferramentas, como "IDEs" e compiladores, usem a estrutura essencial de suporte dele

8.20. Binutils-2.43.1

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

Tempo aproximado de construção: 2,0 UPC

Espaço em disco exigido: 2,7 GB

8.20.1. 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-gold      \
              --enable-ld=default \
              --enable-plugins   \
              --enable-shared    \
              --disable-werror   \
              --enable-64-bit-bfd \
              --enable-new-dtags \
              --with-system-zlib  \
              --enable-default-hash-style=gnu
```

O significado dos novos parâmetros de configuração:

--enable-gold

Constrói o vinculador gold e o instala como ld.gold (juntamente com o vinculador padrão).

--enable-ld=default

Constrói o vinculador bfd original e o instala como ambos ld (o vinculador padrão) e ld.bfd.

--enable-plugins

Habilita suporte de plugin para o vinculador.

--with-system-zlib

Usa a biblioteca zlib instalada em vez de construir a versão incluída.

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-pc-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')
```

Doze testes falham na suíte de teste gold quando as opções `--enable-default-pie` e `--enable-default-ssp` são passadas para o GCC.

Instale o pacote:

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

Remova bibliotecas estáticas inúteis:

```
rm -fv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a
```

8.20.2. Conteúdo do Binutils

Aplicativos instalados: addr2line, ar, as, c++filt, dwp, elfedit, gprof, gprofng, ld, ld.bfd, ld.gold, 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.gold Uma versão reduzida do ld que suporta somente o formato de arquivo objeto elf

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

8.21. 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

8.21.1. 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, [então] bibliotecas genéricas podem ser criadas anexando a opção `--host=none-linux-gnu` ao comando **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 configuração:

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



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:

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

Instale o pacote e a documentação dele:

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

8.21.2. 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++

8.22. MPFR-4.2.1

O pacote MPFR contém funções para matemática de precisão múltipla.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 43 MB

8.22.1. Instalação do MPFR

Prepare MPFR para compilação:

```
./configure --prefix=/usr      \
            --disable-static   \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.1
```

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
```

8.22.2. Conteúdo do MPFR

Bibliotecas instaladas: libmpfr.so

Diretório instalado: /usr/share/doc/mpfr-4.2.1

Descrições Curtas

`libmpfr` Contém funções matemáticas de precisão múltipla

8.23. MPC-1.3.1

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: 22 MB

8.23.1. Instalação do MPC

Prepare MPC para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.3.1
```

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
```

8.23.2. Conteúdo do MPC

Bibliotecas instaladas: libmpc.so

Diretório instalado: /usr/share/doc/mpc-1.3.1

Descrições Curtas

`libmpc` Contém funções matemáticas complexas

8.24. Attr-2.5.2

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: 4,1 MB

8.24.1. Instalação do Attr

Prepare Attr para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.2
```

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
```

8.24.2. Conteúdo do Attr

Aplicativos instalados: attr, getfattr e setfattr

Biblioteca instalada: libattr.so

Diretórios instalados: /usr/include/attr e /usr/share/doc/attr-2.5.2

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

8.25. Acl-2.3.2

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: 6,3 MB

8.25.1. Instalação do Acl

Prepare Acl para compilação:

```
./configure --prefix=/usr      \
            --disable-static   \
            --docdir=/usr/share/doc/acl-2.3.2
```

Compile o pacote:

```
make
```

Os testes do Acl precisam ser executados sobre um sistema de arquivos que suporte controles de acesso, porém não até que o pacote Coreutils tenha sido construído, usando as bibliotecas do Acl. Se desejado, retorne a esse pacote e execute **make check** depois que o pacote Coreutils tiver sido construído.

Instale o pacote:

```
make install
```

8.25.2. Conteúdo do Acl

Aplicativos instalados: chacl, getfacl e setfacl

Biblioteca instalada: libacl.so

Diretórios instalados: /usr/include/acl e /usr/share/doc/acl-2.3.2

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

8.26. Libcap-2.70

O pacote Libcap implementa a interface do espaço de usuário(o) para os recursos POSIX 1003.1e disponíveis em núcleos Linux. Esses recursos particionam o todo poderoso privilégio de root em um conjunto de privilégios distintos.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 2,9 MB

8.26.1. Instalação do Libcap

Impeça 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
```

8.26.2. Conteúdo do Libcap

Aplicativos 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
<code>libcap</code>	Contém as funções de biblioteca para manipular recursos POSIX 1003.1e
<code>libpsx</code>	Contém funções para suportar semântica POSIX para chamadas de sistema associadas com a biblioteca pthread

8.27. Libxcrypt-4.4.36

O pacote Libxcrypt contém uma biblioteca moderna para hash unidirecional de senhas.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 12 MB

8.27.1. Instalação do Libxcrypt

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 configuração:

--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
```

8.27.2. Conteúdo do Libxcrypt

Bibliotecas instaladas: libcrypt.so

Descrições Curtas

`libcrypt` Contém funções para resumir senhas

8.28. Shadow-4.16.0

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

8.28.1. Instalação do Shadow



Importante

Se você tiver instalado o Linux-PAM, você deveria seguir a *página shadow do BLFS* em vez desta página para construir (ou reconstruir ou atualizar) shadow.



Nota

Se você gostaria de reforçar o uso de senhas fortes, [então] recorra a <https://www.linuxfromscratch.org/blfs/view/12.2/postlfs/cracklib.html> para instalar o CrackLib antes de construir o Shadow. Então adicione `--with-libcrack` ao comando **configure** abaixo.

Desabilite a instalação do aplicativo **groups** e as páginas de manual dele, uma vez que o Coreutils fornece uma versão melhor. Também, impeça a instalação de páginas de manual que já foram instaladas no Seção 8.3, “Manpages-6.9.1”:

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

Em vez de usar o método padrão *crypt*, use o método muito mais seguro *YESCRYPT* de encriptação de senha, que também permite senhas com mais de oito (08) caracteres. Também é necessário mudar o local obsoleto `/var/spool/mail` para as caixas de correio de usuário(a) que o Shadow usa por padrão para o local `/var/mail` usado atualmente. E, remova `/bin` e `/sbin` do `PATH`, pois eles são simplesmente links simbólicos para seus contrapartes em `/usr`.



Nota

Se você deseja incluir `/bin` e (ou) `/sbin` no `PATH` por alguma razão, [então] modifique a `PATH` no `.bashrc` depois que o LFS tiver sido construído.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD YESCRYPT:' \
-e 's:/var/spool/mail:/var/mail:' \
-e '/PATH={s@/sbin:@;s@/bin:@}' \
-i etc/login.defs
```



Nota

Se você escolher construir o Shadow com suporte a CrackLib, [então] emita este comando:

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

Prepare o Shadow para compilação:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
--disable-static \
--with-{b,yes}crypt \
--without-libbsd \
--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 usuário(a) mais longo permissível é o de trinta e dois (32) caracteres. Torne o comprimento máximo de um nome de grupo o mesmo.

```
--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
```

8.28.2. Configurando o Shadow

Esse pacote contém utilitários para adicionar, modificar e deletar usuários(as) e grupos; configurar e modificar 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. Se usar suporte ao Shadow, [então] tenha na mente que aplicativos que necessitem verificar senhas (gerenciadores de tela, aplicativos de FTP, processos de segundo plano pop3, etc.) precisam ser conformantes com o Shadow. Isto é, eles precisam ser capazes de funcionar com senhas sombreadas.

Para habilitar senhas sombreadas, execute o seguinte comando:

```
pwconv
```

Para habilitar senhas sombreadas de grupo, execute:

```
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) usuário(a) e um grupo com o mesmo nome que o(a) usuário(a). Por padrão, os números de "ID" de usuário(a) ("UID") e "ID" de grupo ("GID") iniciarão em 1000. Isso significa que, se você não passar parâmetros extras para o "**useradd**", [então] cada usuário(a) será um(a) membro(a) de um grupo único no sistema. Se esse comportamento for indesejável, [então] você precisará passar, ou o parâmetro "`-g`", ou o "`-N`" para o "**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 Links Simbólicos.”

`CREATE_MAIL_SPOOL=yes`

Esse parâmetro faz com que o **useradd** crie um arquivo de caixa de correio para cada novo(a) usuário(a). O **useradd** atribuirá a propriedade de grupo desse arquivo para o grupo `mail` com permissões 0660. Se você, em vez disso, não quisesse criar esses arquivos, [então] emita o seguinte comando:

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

8.28.3. Configurando a Senha do(a) Root

Escolha uma senha para o(a) usuário(a) `root` e configure-a executando:

```
passwd root
```

8.28.4. Conteúdo do Shadow

Aplicativos instalados: `chage`, `chfn`, `chgpaswd`, `chpasswd`, `chsh`, `expiry`, `faillog`, `getsubids`, `gpaswd`, `groupadd`, `groupdel`, `groupmems`, `groupmod`, `grpck`, `grpconv`, `grpunconv`, `login`, `logoutd`, `newgidmap`, `newgrp`, `newuidmap`, `newusers`, `nologin`, `passwd`, `pwck`, `pwconv`, `pwunconv`, `sg` (link para `newgrp`), `su`, `useradd`, `userdel`, `usermod`, `vigr` (link para `vipw`) e `vipw`

Diretórios instalados: `/etc/default` e `/usr/include/shadow`

Bibliotecas instaladas: `libsubid.so`

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) usuário(a) e outra informação
chgpaswd	Usado para atualizar senhas de grupo em modo de lote
chpasswd	Usado para atualizar senhas de usuária(o) em modo de lote
chsh	Usado para mudar um shell de login padrão do(a) usuário(a)
expiry	Verifica e reforça a política atual de expiração de senha
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) usuário(a)
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
groupmems	Permite que um(a) usuário(a) administre a própria lista de filiação de grupo dele/dela sem a exigência de privilégios de superusuário(a).

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 usuárias(os) logar
logoutd	É um processo em segundo plano usado para reforçar restrições sobre horário de logon e portas
newgidmap	É usado para configurar o mapeamento gid de um espaço de nome de usuária(o)
newgrp	É usado para mudar o GID atual durante uma sessão de login
newuidmap	É usado para configurar o mapeamento uid de um espaço de nome de usuária(o)
newusers	É usado para criar ou atualizar uma série inteira de contas de usuárias(os)
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 usuária(o) ou grupo
pwck	Verifica a integridade dos arquivos de senha <code>/etc/passwd</code> e <code>/etc/shadow</code>
pwconv	Cria ou atualiza o arquivo de senha de sombra a partir do arquivo de senha normal
pwunconv	Atualiza <code>/etc/passwd</code> a partir de <code>/etc/shadow</code> e então deleta o último
sg	Executa um comando dado enquanto o GID do(a) usuário(a) estiver configurado para aquele do grupo dado
su	Executa um shell com IDs de usuária(o) e grupo substitutos
useradd	Cria um(a) usuário(a) novo(a) com o nome dado ou atualiza a informação padrão de novo(a) usuário(a)
userdel	Deleta a conta de usuária(o) especificada
usermod	É usado para modificar o nome de login do(a) usuário(a) dada(o), identificação de usuária(o) (UID), shell, grupo inicial, diretório home, etc.
vigr	Edita os arquivos <code>/etc/group</code> ou <code>/etc/gshadow</code>
vipw	Edita os arquivos <code>/etc/passwd</code> ou <code>/etc/shadow</code>
<code>libsubid</code>	Biblioteca para lidar com intervalos subordinados de id para usuárias(os) e grupos

8.29. GCC-14.2.0

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

Tempo aproximado de construção: 46 UPC (com os testes)

Espaço em disco exigido: 6,3 GB

8.29.1. 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 \
             --disable-multilib \
             --disable-bootstrap \
             --disable-fixincludes \
             --with-system-zlib
```

O GCC suporta sete linguagens computacionais, porém os pré-requisitos para a maior parte delas ainda não foram instalados. Veja-se a *página do GCC do Livro BLFS* para instruções a respeito do como construir todas as linguagens suportadas do GCC.

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-fixincludes`

Por padrão, durante a instalação do GCC alguns cabeçalhos de sistema seriam “corrigidos” para serem usados com o GCC. Isso não é necessário para um sistema moderno Linux e potencialmente danoso se um pacote for reinstalado depois de instalar o GCC. Essa chave evita que o GCC “corrija” os cabeçalhos.

`--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.



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
```

Agora remova/corrija várias falhas conhecidas de teste:

```
sed -e '/cpython/d' -i ../gcc/testsuite/gcc.dg/plugin/plugin.exp
sed -e 's/no-pic /&-no-pie /' -i ../gcc/testsuite/gcc.target/i386/pr113689-1.c
sed -e 's/300000/(1|300000)/' -i ../libgomp/testsuite/libgomp.c-c++-common/pr109062.c
sed -e 's/{ target nonpic } //' \
-e '/GOTPCREL/d' -i ../gcc/testsuite/gcc.target/i386/fentryname3.c
```

Teste os resultados como um(a) usuário(a) 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
```

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/12.2/> e <https://gcc.gnu.org/ml/gcc-testresults/>.

Umhas 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 propriedade do diretório do cabeçalho instalado (e o conteúdo dele) está incorreto. Mude a propriedade para o(a) usuário(a) e grupo `root`:

```
chown -v -R root:root \
  /usr/lib/gcc/$(gcc -dumpmachine)/14.2.0/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 -sfv ../../libexec/gcc/$(gcc -dumpmachine)/14.2.0/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(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
readelf -l a.out | grep ': /lib'
```

Não deveriam existir erros e a saída gerada do último comando será (permitindo diferenças específicas de plataforma no nome do vinculador dinâmico):

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

Agora tenha certeza de que nós estamos configurados para usar os arquivos de início corretos:

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

A saída gerada do último comando deveria ser:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../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 de cabeçalho corretos:

```
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/14.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include-fixed
 /usr/include
```

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

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

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

As referências a caminhos que tem 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/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

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 o GCC está usando o vinculador dinâmico correto:

```
grep found dummy.log
```

A saída gerada do último comando deveria ser (permitindo diferenças específicas de 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 como mostrada acima ou não for recebida de jeito nenhum, então alguma coisa está seriamente errada. Investigue e refaça os passos para descobrir onde o problema está e corrija-lo. Quaisquer problemas deveriam ser resolvidos antes de continuar com o processo.

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

```
rm -v dummy.c 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
```

8.29.2. Conteúdo do GCC

Aplicativos 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}, libcc1.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-14.2.0

Descrições Curtas

c++	O compilador C++
cc	O compilador C
cpp	O preprocessador C; é usado pelo compilador para expandir as diretivas <code>#include</code> , <code>#define</code> 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
<code>libasan</code>	A biblioteca de tempo de execução Address Sanitizer
<code>libatomic</code>	Biblioteca de tempo de execução integrada atômica do GCC
<code>libcc1</code>	Uma biblioteca que permite ao "GDB" fazer uso do "GCC"
<code>libgcc</code>	Contém suporte de tempo de execução para o gcc
<code>libgcov</code>	Essa biblioteca é vinculada a um aplicativo quando o GCC for instruído a habilitar criação de perfil
<code>libgomp</code>	Implementação GNU da API OpenMP para programação paralela de memória compartilhada multiplataforma em C/C++ e Fortran
<code>libhwasan</code>	A biblioteca de tempo de execução do Address Sanitizer assistida por hardware
<code>libitm</code>	A biblioteca de memória transacional GNU
<code>liblsan</code>	A biblioteca de tempo de execução Leak Sanitizer
<code>liblto_plugin</code>	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++

`libtsan`

A biblioteca de tempo de execução Thread Sanitizer

`libubsan`

A biblioteca de tempo de execução Undefined Behavior Sanitizer

8.30. Ncurses-6.5

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: 46 MB

8.30.1. 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 configuração:

--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.6.5` 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 o arquivo da biblioteca corretamente usando o comando `install` (o cabeçalho `curses.h` também é editado para garantir que a ABI de caracteres largos seja usada como fizemos em Seção 6.3, “Ncurses-6.5”):

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.5 /usr/lib
rm -v dest/usr/lib/libncursesw.so.6.5
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
     -i dest/usr/include/curses.h
cp -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, [então] instale a documentação do Ncurses:

```
cp -v -R doc -T /usr/share/doc/ncurses-6.5
```



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, [então] 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
```

8.30.2. Conteúdo do Ncurses

Aplicativos instalados:	captainfo (link para tic), clear, infocmp, infotocap (link para tic), ncursesw6-config, reset (link para tset), tabs, tic, toe, tput e tset
Bibliotecas instaladas:	libcurses.so (link simbólico), libform.so (link simbólico), libformw.so, libmenu.so (link simbólico), libmenuw.so, libncurses.so (link simbólico), libncursesw.so, libncurses++w.so, libpanel.so (link simbólico) e libpanelw.so,
Diretórios instalados:	/usr/share/tabset, /usr/share/terminfo e /usr/share/doc/ncurses-6.5

Descrições Curtas

captainfo	Converte uma descrição termcap em uma descrição terminfo
clear	Limpa a tela, se possível
infocmp	Compara ou imprime descrições terminfo
infotocap	Converte uma descrição terminfo em uma descrição termcap
ncursesw6-config	Fornecer informação de configuração para o ncurses
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 terminfo que traduz um arquivo terminfo do formato fonte para o formato binário necessário para as rotinas de biblioteca do ncurses [Um arquivo terminfo 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

8.31. Sed-4.9

O pacote Sed contém um editor de fluxo.

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 30 MB

8.31.1. 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 -d -m755 /usr/share/doc/sed-4.9
install -m644 doc/sed.html /usr/share/doc/sed-4.9
```

8.31.2. Conteúdo do Sed

Aplicativo instalado: sed
Diretório instalado: /usr/share/doc/sed-4.9

Descrições Curtas

sed Filtra e transforma arquivos de texto em uma passagem única

8.32. 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: 6,7 MB

8.32.1. 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
```

8.32.2. Conteúdo do Psmisc

Aplicativos 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

8.33. Gettext-0.22.5

O pacote Gettext contém utilitários para internacionalização e localização. Eles permitem que aplicativos sejam compilados com Suporte ao Idioma Nativo (Native Language Support - NLS), habilitando-os a emitir mensagens no idioma nativo do(a) usuário(a).

Tempo aproximado de 1,6 UPC

construção:

Espaço em disco exigido: 260 MB

8.33.1. Instalação do Gettext

Prepare o Gettext para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.22.5
```

Compile o pacote:

```
make
```

Para testar os resultados (isso toma um tempo longo, em torno de 3 UPCs), emita:

```
make check
```

Instale o pacote:

```
make install
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

8.33.2. Conteúdo do Gettext

Aplicativos 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-0.22.5, /usr/share/gettext e /usr/share/gettext-0.22.5

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) usuário(a) 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 <code>.po</code> 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	Extrai 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 <code>.po</code> , inicializando a meta informação com valores oriundos do ambiente do(a) usuário(a)
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	Extrai as linhas de mensagem traduzíveis dos arquivos fonte fornecidos para fazer o primeiro modelo de tradução
<code>libasprintf</code>	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 <code><string></code> e os fluxos <code><iostream></code>
<code>libgettextlib</code>	Contém rotinas comuns usadas pelos vários aplicativos do Gettext; elas não são destinadas para uso geral
<code>libgettextpo</code>	Usada para escrever aplicativos especializados que processam arquivos <code>.po</code> ; essa biblioteca é usada quando os aplicativos padrão embarcados com o Gettext (tais como msgcomm , msgcmp , msgattrib e msgen) não bastarão
<code>libgettextsrc</code>	Fornecer rotinas comuns usadas pelos vários aplicativos do Gettext; elas não são destinadas para uso geral
<code>libtextstyle</code>	Biblioteca de estilização de texto
<code>preloadable_libintl</code>	Uma biblioteca, destinada a ser usada por <code>LD_PRELOAD</code> que auxilia a <code>libintl</code> a registrar mensagens não traduzidas

8.34. Bison-3.8.2

O pacote Bison contém um gerador de analisador.

Tempo aproximado de construção: 2,2 UPC

Espaço em disco exigido: 62 MB

8.34.1. 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 (cerca de 5,5 UPCs), emita:

```
make check
```

Instale o pacote:

```
make install
```

8.34.2. Conteúdo do Bison

Aplicativos 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

8.35. Grep-3.11

O pacote Grep contém aplicativos para procura ao longo do conteúdo de arquivos.

Tempo aproximado de 0,4 UPC

construção:

Espaço em disco exigido: 39 MB

8.35.1. 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
```

8.35.2. Conteúdo do Grep

Aplicativos 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

8.36. Bash-5.2.32

O pacote Bash contém o Bourne-Again SHell.

Tempo aproximado de 1,4 UPC

construção:

Espaço em disco exigido: 52 MB

8.36.1. Instalação do Bash

Prepare o Bash para compilação:

```
./configure --prefix=/usr \
            --without-bash-malloc \
            --with-installed-readline \
            bash_cv_strtold_broken=no \
            --docdir=/usr/share/doc/bash-5.2.32
```

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) usuário(a) `tester` consegue escrever na árvore dos fontes:

```
chown -R tester .
```

A suíte de teste desse pacote é projetada para ser executada como um(a) usuário(a) 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) usuário(a) `tester`:

```
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. Um teste chamado `run-builtins` é conhecido por falhar em algumas distribuições anfitriãs com uma diferença na primeira linha da saída gerada.

Instale o pacote:

```
make install
```

Execute o aplicativo recém compilado **bash** (substituindo o que está sendo executado atualmente):

```
exec /usr/bin/bash --login
```

8.36.2. Conteúdo do Bash

Aplicativos instalados: bash, bashbug e sh ([link para bash](#))

Diretório instalado: /usr/include/bash, /usr/lib/bash e /usr/share/doc/bash-5.2.32

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 script de shell para ajudar o(a) usuário(a) a compor e enviar relatórios de defeitos formatados padrão 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

8.37. Libtool-2.4.7

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 0,8 UPC

construção:

Espaço em disco exigido: 45 MB

8.37.1. Instalação do Libtool

Prepare Libtool para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make -k check
```

Cinco testes são conhecidos por falharem no ambiente de construção do LFS, devido a uma dependência circular, porém esses testes passam se verificados novamente depois que o "automake" tiver sido instalado. Adicionalmente, com o "grep-3.8" ou mais recente, dois testes deflagrarão um aviso para expressões regulares não "POSIX" e falharão.

Instale o pacote:

```
make install
```

Remova uma biblioteca estática inútil:

```
rm -fv /usr/lib/libltdl.a
```

8.37.2. Conteúdo do Libtool

Aplicativos 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

8.38. GDBM-1.24

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: menos que 0,1 UPC

Espaço em disco exigido: 13 MB

8.38.1. 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
```

8.38.2. Conteúdo do GDBM

Aplicativos instalados: gdbm_dump, gdbm_load e gdbmtool

Bibliotecas instaladas: libgdbm.so e libgdbm_compat.so

Descrições Curtas

<code>gdbm_dump</code>	Despeja uma base de dados GDBM para um arquivo
<code>gdbm_load</code>	Recria uma base de dados GDBM a partir de um arquivo de despejo
<code>gdbmtool</code>	Testa e modifica uma base de dados GDBM
<code>libgdbm</code>	Contém funções para manipular uma base de dados com hash
<code>libgdbm_compat</code>	Biblioteca de compatibilidade contendo funções DBM mais antigas

8.39. Gperf-3.1

Gperf gera uma função de hash perfeita a partir de um conjunto de chaves.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 6,1 MB

8.39.1. Instalação do Gperf

Prepare Gperf para compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

Compile o pacote:

```
make
```

Os testes são conhecidos por falharem se executar múltiplos testes simultâneos (opção -j maior que 1). Para testar os resultados, emita:

```
make -j1 check
```

Instale o pacote:

```
make install
```

8.39.2. Conteúdo do Gperf

Aplicativo instalado: gperf

Diretório instalado: /usr/share/doc/gperf-3.1

Descrições Curtas

gperf Gera um hash perfeito a partir de um conjunto de chaves

8.40. Expat-2.6.2

O pacote Expat contém uma biblioteca C orientada a fluxo para analisar XML.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 13 MB

8.40.1. Instalação do Expat

Prepare Expat para compilação:

```
./configure --prefix=/usr \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.6.2
```

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-2.6.2
```

8.40.2. Conteúdo do Expat

Aplicativo instalado: xmlwf

Bibliotecas instaladas: libexpat.so

Diretório instalado: /usr/share/doc/expat-2.6.2

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

8.41. Inetutils-2.5

O pacote Inetutils contém aplicativos para operação interativa básica de dispositivos via rede de comunicação.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 35 MB

8.41.1. 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
```

8.41.2. Conteúdo do Inetutils

Aplicativos instalados: dnsdomainname, ftp, ifconfig, hostname, ping, ping6, talk, telnet, tftp e traceroute

Descrições Curtas

dnsdomainname	Mostra o nome de domínio DNS do sistema
ftp	É o aplicativo do protocolo de transferência de arquivos
hostname	Informa ou configura o nome do dispositivo
ifconfig	Gerencia interfaces da rede de comunicação
ping	Envia pacotes de solicitação de echo e informa quanto tempo as respostas demoram
ping6	Uma versão do ping para redes de comunicação IPv6
talk	É usado para conversar com outro(a) usuário(a)
telnet	Uma interface para o protocolo TELNET
tftp	Um aplicativo de transferência de arquivos trivial
traceroute	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

8.42. Less-661

O pacote Less contém um visualizador de arquivos de texto.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 14 MB

8.42.1. 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
```

8.42.2. Conteúdo do Less

Aplicativos instalados: less, lessecho e lesskey

Descrições Curtas

- less** Um visualizador de arquivos ou paginador; ele exibe o conteúdo do arquivo dado, permitindo que o(a) usuário(a) role, encontre sequências de caracteres e pule para marcas
- lessecho** Necessário para expandir meta caracteres, tais como `*` e `?`, em nomes de arquivos em sistemas Unix
- lesskey** Usado para especificar os atalhos de tecla para o **less**

8.43. Perl-5.40.0

O pacote Perl contém o Practical Extraction and Report Language.

Tempo aproximado de construção: 1,4 UPC

Espaço em disco exigido: 245 MB

8.43.1. 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/perl5/5.40/core_perl \
-D archlib=/usr/lib/perl5/5.40/core_perl \
-D sitelib=/usr/lib/perl5/5.40/site_perl \
-D sitearch=/usr/lib/perl5/5.40/site_perl \
-D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
-D vendorarch=/usr/lib/perl5/5.40/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 (aproximadamente 11 UPCs), emita:

```
TEST_JOBS=$(nproc) make test_harness
```

Instale o pacote e limpe:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.43.2. Conteúdo do Perl

Aplicativos instalados:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.40.0 (link rígido para perl), perlbug, perldoc, perlivp, 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) usuário(a) 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
perl5.40.0	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
perlivp	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

8.44. XML::Parser-2.47

O módulo XML::Parser é uma interface Perl para o analisador de XML do James Clark, Expat.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 2,4 MB

8.44.1. Instalação do XML::Parser

Prepare XML::Parser para compilação:

```
perl Makefile.PL
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make test
```

Instale o pacote:

```
make install
```

8.44.2. Conteúdo do XML::Parser

Módulo instalado: Expat.so

Descrições Curtas

`Expat` Fornece a interface Perl Expat

8.45. Intltool-0.51.0

O Intltool é uma ferramenta de internacionalização usada para extrair sequências de caracteres traduzíveis a partir de arquivos fonte.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 1,5 MB

8.45.1. Instalação do Intltool

Primeiro, corrija um aviso que é causado por perl-5.22 e posteriores:

```
sed -i 's:\\\\${:\\\\$\\{: ' intltool-update.in
```



Nota

A expressão regular acima parece incomum por causa de todas as contra barras. O que ela faz é adicionar uma contra barra antes do carácter abre chave na sequência '\\${' resultando em '\\\${'.

Prepare Intltool para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
install -v -Dm644 doc/I18N-HOWTO /usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

8.45.2. Conteúdo do Intltool

Aplicativos instalados: intltool-extract, intltool-merge, intltool-prepare, intltool-update e intltoolize

Diretórios instalados: /usr/share/doc/intltool-0.51.0 e /usr/share/intltool

Descrições Curtas

intltoolize	Prepara um pacote para usar intltool
intltool-extract	Gera arquivos de cabeçalho que podem ser lidos por gettext
intltool-merge	Mescla sequência de caracteres traduzidos em vários tipos de arquivos
intltool-prepare	Atualiza arquivos pot e os mescla com arquivos de tradução
intltool-update	Atualiza os arquivos de modelo po e os mescla com as traduções

8.46. Autoconf-2.72

O pacote Autoconf contém aplicativos para produzir scripts de shell que conseguem configurar automaticamente código fonte.

Tempo aproximado de menos que 0,1 UPC (cerca de 1,8 UPC com os testes)

construção:

Espaço em disco exigido: 25 MB

8.46.1. 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
```

8.46.2. Conteúdo do Autoconf

Aplicativos 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 aplicativos; ele imprime os identificadores que o pacote usa em condicionais de pré-processador C [Se um pacote já tenha sido configurado para ter alguma portabilidade, [então] esse aplicativo pode ajudar a determinar o que o configure precisa verificar. Ele também consegue preencher lacunas em um arquivo <code>configure.in</code> gerado pelo autoscan].

8.47. Automake-1.17

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: 121 MB

8.47.1. Instalação do Automake

Prepare Automake para compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.17
```

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
```

8.47.2. Conteúdo do Automake

Aplicativos instalados: `aclocal`, `aclocal-1.17` (vinculado rigidamente com `aclocal`), `automake` e `automake-1.17` (vinculado rigidamente com `automake`)

Diretórios instalados: `/usr/share/aclocal-1.17`, `/usr/share/automake-1.17` e `/usr/share/doc/automake-1.17`

Descrições Curtas

aclocal	Gera arquivos <code>aclocal.m4</code> baseados no conteúdo dos arquivos <code>configure.in</code>
aclocal-1.17	Um link rígido para aclocal
automake	Uma ferramenta para gerar automaticamente arquivos <code>Makefile.in</code> a partir de arquivos <code>Makefile.am</code> [Para criar todos os arquivos <code>Makefile.in</code> para um pacote, execute esse aplicativo no diretório de nível superior. Escaneando o arquivo <code>configure.in</code> , ele automaticamente encontra cada arquivo <code>Makefile.am</code> apropriado e gera o arquivo <code>Makefile.in</code> correspondente].
automake-1.17	Um link rígido para automake

8.48. OpenSSL-3.3.1

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 1,7 UPC

construção:

Espaço em disco exigido: 883 MB

8.48.1. 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:

```
HARNESS_JOBS=$(nproc) make test
```

Um teste, 30-test_afalg.t, é conhecido por falhar se o núcleo do anfitrião não tiver `CONFIG_CRYPT_USER_API_SKCIPHER` habilitado ou não tiver quaisquer opções fornecendo um AES com implementação CBC (por exemplo, a combinação de `CONFIG_CRYPT_AES` e `CONFIG_CRYPT_CBC`; ou `CONFIG_CRYPT_AES_NI_INTEL` se a CPU suportar AES-NI) habilitado. Se falhar, pode ser seguramente ignorado.

Instale o pacote:

```
sed -i '/INSTALL LIBS/s/libcrypto.a libssl.a//' Makefile
make 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-3.3.1
```

Se desejado, instale alguma documentação adicional:

```
cp -vfr doc/* /usr/share/doc/openssl-3.3.1
```



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. Leia-se as entradas relacionadas em Seção 8.2.1, “Problemas de Atualização” para detalhes.

8.48.2. Conteúdo do OpenSSL

Aplicativos 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-3.3.1`

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> "
<code>libcrypto.so</code>	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
<code>libssl.so</code>	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> "

8.49. Kmod-33

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: 11 MB

8.49.1. Instalação do Kmod

Prepare Kmod para compilação:

```
./configure --prefix=/usr \
            --sysconfdir=/etc \
            --with-openssl \
            --with-xz \
            --with-zstd \
            --with-zlib \
            --disable-manpages
```

O significado das opções do configure:

--with-openssl

Essa opção habilita o Kmod a lidar com assinaturas PKCS7 para módulos de núcleo.

--with-xz, --with-zlib e --with-zstd

Essas opções habilitam o Kmod a lidar com módulos comprimidos de núcleo.

--disable-manpages

Essa opção desabilita gerar as páginas de manual, o que exige um programa externo.

Compile o pacote:

```
make
```

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.

Instale o pacote e recrie alguns links simbólicos para compatibilidade com Module-Init-Tools (o pacote que anteriormente manuseava os módulos de núcleo do Linux). O sistema de construção criará todos esses links simbólicos em `/usr/bin`, mas queremos somente **lsmod** lá e todos os outros links simbólicos em `/usr/sbin` em vez disso:

```
make install

for target in depmod insmod modinfo modprobe rmmod; do
    ln -sfv ../bin/kmod /usr/sbin/$target
    rm -fv /usr/bin/$target
done
```

8.49.2. Conteúdo do Kmod

Aplicativos 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
<code>libkmod</code>	Essa biblioteca é usada por outros aplicativos para carregar e descarregar módulos de núcleo

8.50. Libelf originário do Elfutils-0.191

Libelf é uma biblioteca para lidar com arquivos ELF (Executable and Linkable Format).

Tempo aproximado de construção: 0,3 UPC

Espaço em disco exigido: 127 MB

8.50.1. Instalação do Libelf

Libelf é parte do pacote elfutils-0.191. Use o arquivo elfutils-0.191.tar.bz2 como o tarball fonte.

Prepare Libelf para compilação:

```
./configure --prefix=/usr \
            --disable-debuginfod \
            --enable-libdebuginfod=dummy
```

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

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-3.4.6

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: 11 MB

8.51.1. Instalação do Libffi



Nota

Semelhante a GMP, Libffi constrói com otimizações específicas para o processador em uso. Se construir para outro sistema, [então] mude o valor do parâmetro `--with-gcc-arch=` no seguinte comando para um nome de arquitetura completamente implementada pela CPU naquele sistema. Se isso não for feito, [então] todos os aplicativos que se vincularem a `libffi` deflagrarão Erros de Operação Ilegal.

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 optimize 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, vejam-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. Python-3.12.5

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 2,2 UPC

construção:

Espaço em disco exigido: 530 MB

8.52.1. Instalação do Python 3

Prepare o Python para compilação:

```
./configure --prefix=/usr \
            --enable-shared \
            --with-system-expat \
            --enable-optimizations
```

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 elemento de processamento da CPU) deveria ser suficiente. Alguns testes são instáveis, de forma que a suíte de teste automaticamente reexecutará os testes falhos. Se um teste falhou, mas então passou quando reexecutado, ele deveria ser considerado como passado.

Instale o pacote:

```
make install
```

Nós usamos o comando **pip3** para instalar os aplicativos e módulos do Python 3 para todos(as) os(as) usuários(as) como `root` em vários lugares neste livro. Isso conflita com a recomendação dos(as) desenvolvedores(as) do Python: instalar pacotes no ambiente virtual ou no diretório home de um(a) usuário(a) regular (executando **pip3** como esse(a) usuário(a)). Um aviso multi linhas é deflagrado sempre que **pip3** for emitido pelo(a) usuário(a) `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 comunicação, um aviso diferente será emitido, informando para o(a) usuário(a) 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 forma 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) usuário(a) `root` (a menos que seja para um ambiente virtual do Python). Executar um **pip3 install** como um(a) usuário(a) não `root` possivelmente aparente funcionar, porém causará o módulo instalado ficar inacessível por outros(as) usuários(as).

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-3.12.5/html
tar --no-same-owner \
-xvf ../python-3.12.5-docs-html.tar.bz2
cp -R --no-preserve=mode python-3.12.5-docs-html/* \
/usr/share/doc/python-3.12.5/html
```

O significado dos comandos de instalação da documentação:

`--no-same-owner` ("tar") e `--no-preserve=mode` ("cp")

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) e os arquivos teriam permissões restritivas.

8.52.2. Conteúdo do Python 3

Aplicativos instalados: 2to3, idle3, pip3, pydoc3, python3 e python3-config
Biblioteca instalada: libpython3.12.so e libpython3.so
Diretórios instalados: /usr/include/python3.12, /usr/lib/python3 e /usr/share/doc/python-3.12.5

Descrições Curtas

2to3 é um aplicativo Python que lê código fonte Python 2.x e aplica uma série de correções para transformá-lo em código Python 3.x válido

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

8.53. Flit-Core-3.9.0

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: 1,6 MB

8.53.1. 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 --no-user --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 do repositório de pacotes online (PyPI). Se os pacotes fossem instalados na ordem correta, o pip não precisaria buscar nenhum arquivo em primeiro lugar; essas opções adicionam alguma segurança em caso de erro do(a) usuário(a).

--find-links dist

Instrui o pip a procurar por arquivamentos wheel no diretório `dist`.

8.53.2. Conteúdo do Flit-Core

Diretório instalado: `/usr/lib/python3.12/site-packages/flit_core` e `/usr/lib/python3.12/site-packages/flit_core-3.9.0.dist-info`

8.54. Wheel-0.44.0

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: 1,5 MB

8.54.1. 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
```

8.54.2. Conteúdo do Wheel

Aplicativo instalado: wheel

Diretórios instalados: /usr/lib/python3.12/site-packages/wheel e /usr/lib/python3.12/site-packages/wheel-0.44.0.dist-info

Descrições Curtas

wheel é um utilitário para desempacotar, empacotar ou converter arquivamentos roda

8.55. Setuptools-72.2.0

"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: 35 MB

8.55.1. 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
```

8.55.2. Conteúdo do “Setuptools”

Diretório instalado: /usr/lib/python3.12/site-packages/_distutils_hack, /usr/lib/python3.12/site-packages/pkg_resources, /usr/lib/python3.12/site-packages/setuptools e /usr/lib/python3.12/site-packages/setuptools-72.2.0.dist-info

8.56. Ninja-1.12.1

Ninja é um sistema de construção pequeno com um foco em velocidade.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 37 MB

8.56.1. 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) usuário(a) 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
```

O significado da opção de construção:

`--bootstrap`

Esse parâmetro força Ninja a reconstruir ele próprio para o sistema atual.

Os testes do pacote não podem executar no ambiente `chroot`. Eles exigem `cmake`.

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
```

8.56.2. Conteúdo do Ninja

Aplicativos instalados: ninja

Descrições Curtas

ninja é o sistema de construção Ninja

8.57. Meson-1.5.1

Meson é um sistema de construção de código fonte aberto projetado para ser ambos extremamente rápido e tão amigável para o(a) usuário(a) quanto possível.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 43 MB

8.57.1. 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`.

8.57.2. Conteúdo do Meson

Aplicativos instalados: meson

Diretório instalado: /usr/lib/python3.12/site-packages/meson-1.5.1.dist-info e /usr/lib/python3.12/site-packages/mesonbuild

Descrições Curtas

meson Um sistema de construção de alta produtividade

8.58. Coreutils-9.5

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

Tempo aproximado de 1,1 UPC

construção:

Espaço em disco exigido: 173 MB

8.58.1. 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-9.5-i18n-2.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 -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
    --prefix=/usr \
    --enable-no-install-program=kill,uptime
```

O significado das opções do configure:

autoreconf

O remendo para internacionalização modificou o sistema de construção, de forma que os arquivos de configuração precisam ser regenerados.

```
FORCE_UNSAFE_CONFIGURE=1
```

Essa variável de ambiente permite que o pacote seja construído pelo(a) usuário(a) `root`.

```
--enable-no-install-program=kill,uptime
```

O propósito dessa chave é o de impedir que o Coreutils instale aplicativos que serão instalados por outros pacotes.

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 usuário(a) `root`:

```
make NON_ROOT_USERNAME=tester check-root
```

Nós vamos executar o resto dos testes como o(a) usuário(a) `tester`. Certos testes exigem que o(a) usuário(a) seja um(a) membro(a) de mais que um grupo. Portanto, para que esses testes não sejam pulados, adicione um grupo temporário e torne o(a) usuário(a) `tester` uma parte dele:

```
groupadd -g 102 dummy -U tester
```

Corrija algumas das permissões, de modo que o(a) usuário(a) não `root` possa 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
```

Dois testes, `tests/cp/preserve-mode.sh` e `tests/mv/acl.sh`, são conhecidos por falharem no ambiente chroot, mas passam em um sistema completo.

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
```

8.58.2. Conteúdo do Coreutils

Aplicativos instalados: `[], b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami` 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

chcon Muda contexto de segurança para arquivos e diretórios

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 propriedade de usuário(a) e (ou) de grupo de arquivos e dos 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 <code>LS_COLOR</code> 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) usuário(a)
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 usuária(o), ID de grupo e as associações de grupo do(a) usuário(a) atual ou usuária(o) especificada(o)
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) usuário(a) 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 portáveis
pinky	É um cliente de dedo leve; ele informa alguma informação a respeito das(os) usuárias(os) dadas(os)
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
runcon	Executa um comando com contexto de segurança especificado
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 das(os) usuárias(os) atualmente logadas(os)
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 usuária(o) associado com o ID efetivo de usuária(o) atual
yes	Repetidamente produz y ou uma dada sequência de caracteres, até ser eliminado
<code>libstdbuf</code>	Biblioteca usada por stdbuf

8.59. Check-0.15.2

Check é uma estrutura de teste de unidade para C.

Tempo aproximado de construção: 0,1 UPC (cerca de 2,1 UPC com os testes)

Espaço em disco exigido: 11 MB

8.59.1. Instalação do Check

Prepare Check para compilação:

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

Construa o pacote:

```
make
```

A compilação agora está completa. Para executar a suíte de teste do Check, emita o seguinte comando:

```
make check
```

Instale o pacote:

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.59.2. Conteúdo do Check

Aplicativo instalado: checkmk

Biblioteca instalada: libcheck.so

Descrições Curtas

checkmk Script awk para gerar testes de unidade C para uso com a estrutura de teste de unidade do Check

libcheck.so Contém funções que permitem que o Check seja chamado a partir de um aplicativo de teste

8.60. Diffutils-3.10

O pacote Diffutils contém aplicativos que mostram as diferenças entre arquivos ou diretórios.

Tempo aproximado de 0,3 UPC

construção:

Espaço em disco exigido: 42 MB

8.60.1. 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
```

8.60.2. Conteúdo do Diffutils

Aplicativos 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

8.61. Gawk-5.3.0

O pacote Gawk contém aplicativos para manipular arquivos de texto.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 42 MB

8.61.1. 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-5.3.0
make install
```

O significado do comando:

rm -f /usr/bin/gawk-5.3.0

O sistema de construção não recriará o link rígido `gawk-5.3.0` se ele já existir. Remova-o para garantir que o link rígido anterior instalado na Seção 6.9, “Gawk-5.3.0” 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:

```
mkdir -pv /usr/share/doc/gawk-5.3.0
cp -v doc/{awkforai.txt,*.eps,pdf,jpg} /usr/share/doc/gawk-5.3.0
```

8.61.2. Conteúdo do Gawk

Aplicativos instalados: awk (link para gawk), gawk e awk-5.3.0

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-5.3.0

Descrições Curtas

awk Um link para **gawk**

gawk Um aplicativo para manipular arquivos de texto; é a implementação GNU do **awk**

gawk-5.3.0 Um link rígido para **gawk**

8.62. Findutils-4.10.0

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 0,7 UPC

construção:

Espaço em disco exigido: 62 MB

8.62.1. 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"
```

Instale o pacote:

```
make install
```

8.62.2. Conteúdo do Findutils

Aplicativos 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

8.63. Groff-1.23.0

O pacote Groff contém aplicativos para processar e formatar texto e imagens.

Tempo aproximado de construção: 0,2 UPC

Espaço em disco exigido: 107 MB

8.63.1. Instalação do Groff

Groff espera que a variável de ambiente `PAGE` contenha o tamanho padrão de papel. Para usuárias(os) 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 ou “A4” ou “letter” para o arquivo `/etc/papersize`.

Prepare Groff para compilação:

```
PAGE=<tamanho_papel> ./configure --prefix=/usr
```

Construa o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.63.2. Conteúdo do Groff

Aplicativos instalados: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit e troff

Diretórios instalados: /usr/lib/groff, /usr/share/doc/groff-1.23.0 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 preprocessador 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(a) para o sistema de formatação de documentos groff ; ele executa o aplicativo 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 <code>-e</code> , <code>-man</code> , <code>-me</code> , <code>-mm</code> , <code>-ms</code> , <code>-p</code> , <code>-s</code> e <code>-t</code> 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 preprocessador 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 <code>.pfb</code> 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

8.64. GRUB-2.12

O pacote GRUB contém o GRand Unified Bootloader.

Tempo aproximado de 0,3 UPC

construção:

Espaço em disco exigido: 166 MB

8.64.1. Instalação do GRUB



Nota

Se o teu sistema tem suporte UEFI e você deseja inicializar o LFS com UEFI, você precisa instalar o GRUB com suporte UEFI (e as dependências dele) seguindo as instruções na *página BLFS*. Você pode ignorar esse pacote ou instalar esse pacote e o pacote GRUB para UEFI do BLFS sem conflito (a página BLFS fornece instruções para ambos os casos).



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.

Adicione um arquivo ausente a partir do "tarball" de lançamento:

```
echo depends bli part_gpt > grub-core/extra_deps.lst
```

Prepare GRUB para compilação:

```
./configure --prefix=/usr          \  
            --sysconfdir=/etc      \  
            --disable-efiemu       \  
            --disable-werror
```

O significado das novas opções de configuração:

`--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 aplicativos 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  
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Tornar teu sistema LFS inicializável com GRUB será discutido no Seção 10.4, “Usando o GRUB para Configurar o Processo de Inicialização.”

8.64.2. Conteúdo do GRUB

Aplicativos instalados:	grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup e grub-syslinux2cfg
Diretórios instalados:	/usr/lib/grub, /etc/grub.d, /usr/share/grub e /boot/grub (quando grub-install for primeiro executado)

Descrições Curtas

grub-bios-setup	É um aplicativo auxiliar para grub-install
grub-editenv	Uma ferramenta para editar o bloco ambiente
grub-file	Verifica para ver se o arquivo dado é do tipo especificado
grub-fstest	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 sua unidade
grub-kbdcomp	É um script que converte um esquema xkb em um reconhecido pelo GRUB
grub-macbless	É o bless ao estilo Mac para os sistemas de arquivos HFS ou HFS+ (bless é peculiar às máquinas Apple; torna um dispositivo inicializável)
grub-menulst2cfg	Converte um <code>menu.lst</code> do GRUB Legacy em um <code>grub.cfg</code> para uso com GRUB 2
grub-mkconfig	Gera um arquivo <code>grub.cfg</code>
grub-mkimage	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 aplicativo auxiliar que imprime o caminho para um dispositivo do GRUB
grub-probe	Sonda informação de dispositivo para um caminho ou dispositivo dado
grub-reboot	Configura a entrada padrão de inicialização para o GRUB para a próxima inicialização somente
grub-render-label	Renderiza <code>.disk_label</code> da Apple para Macs da Apple
grub-script-check	Verifica script 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 aplicativo auxiliar para <code>grub-setup</code>
grub-syslinux2cfg	Transforma um arquivo de configuração <code>syslinux</code> para o formato <code>grub.cfg</code>

8.65. Gzip-1.13

O pacote Gzip contém aplicativos para comprimir e descomprimir arquivos.

Tempo aproximado de 0,3 UPC

construção:

Espaço em disco exigido: 21 MB

8.65.1. 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
```

8.65.2. Conteúdo do Gzip

Aplicativos 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

8.66. IPRoute2-6.10.0

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: 17 MB

8.66.1. 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:

```
mkdir -pv /usr/share/doc/iproute2-6.10.0
cp -v COPYING README* /usr/share/doc/iproute2-6.10.0
```

8.66.2. Conteúdo do IPRoute2

Aplicativos 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/iproute2-6.10.0

Descrições Curtas

bridge Configura pontes de redes de comunicação

ctstat Utilitário de situação de conexão

genl Estrutura genérica de interação direta com o(a) usuário(a) do utilitário de link de rede de comunicação

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 usuários(as) olhem para o estado de dispositivos e façam mudanças

ip addr permite que usuários(as) olhem para endereços e propriedades deles, adicionem novos endereços e deletem os antigos

ip neighbor permite que usuários(as) olhem para vínculos de vizinho e propriedades deles, adicionem novas entradas de vizinho e deletem as antigas

ip rule permite que usuários(as) olhem para as políticas de roteamento e as mudem

ip route permite que usuários(as) olhem para a tabela de roteamento e mudem regras da tabela de roteamento

ip tunnel permite que usuários(as) olhem para os tuneis IP e propriedades deles e as mudem

ip maddr permite que usuários(as) olhem para os endereços multicast e propriedades deles e as mudem

ip mroute permite que usuários(as) configurem, mudem ou deletem o roteamento multicast

ip monitor permite que usuários(as) 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 `/proc/net/rt_acct`
- 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 usuários(as) configurem a disciplina de enfileiramento
- tc class** permite que usuários(as) configurem classes baseadas no agendamento da disciplina de enfileiramento
- tc filter** permite que usuários(as) configurem a filtragem de pacote QoS/CoS
- tc monitor** pode ser usado para visualizar mudanças feitas para o Traffic Control no núcleo.

8.67. Kbd-2.6.4

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: 34 MB

8.67.1. 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-2.6.4-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. Usuários(as) 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-2.6.4
```

8.67.2. Conteúdo do Kbd

Aplicativos instalados: chvt, dealloct, 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-2.6.4 e /usr/share/unimaps

Descrições Curtas

chvt	Muda o terminal virtual de primeiro plano
dealloct	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 aplicativo obsoleto que costumava carregar uma tabela de mapeamento de caractere de saída gerada definida pelo(a) usuário(a) para dentro do controlador de console; isso é feito agora 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 aplicativo, a menos que seu arquivo de mapa de tecla esteja na codificação ISO-8859-1. Para outras codificações, esse utilitário produz resultados incorretos].

unicode_stop

Reverte teclado e console do modo UNICODE

8.68. Libpipeline-1.5.7

O pacote Libpipeline contém uma biblioteca para manipular pipelines de subprocessos de uma maneira flexível e conveniente.

Tempo aproximado de 0,1 UPC

construção:

Espaço em disco exigido: 9,7 MB

8.68.1. Instalação do Libpipeline

Prepare Libpipeline para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.68.2. Conteúdo do Libpipeline

Biblioteca instalada: libpipeline.so

Descrições Curtas

`libpipeline` Essa biblioteca é usada para construir seguramente pipelines entre subprocessos

8.69. 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 0,7 UPC

construção:

Espaço em disco exigido: 13 MB

8.69.1. 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
```

8.69.2. 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

8.70. Patch-2.7.6

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: 12 MB

8.70.1. 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
```

8.70.2. 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 aplicativo **diff**. Aplicando essas diferenças aos arquivos originais, **patch** cria as versões remendadas).

8.71. 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 0,7 UPC

construção:

Espaço em disco exigido: 43 MB

8.71.1. Instalação do Tar

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) usuário(a) `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
```

8.71.2. Conteúdo do Tar

Aplicativos 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

8.72. Texinfo-7.1

O pacote Texinfo contém aplicativos para leitura, escrita e conversão de páginas info.

Tempo aproximado de 0,3 UPC

construção:

Espaço em disco exigido: 139 MB

8.72.1. 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
```

8.72.2. Conteúdo do Texinfo

Aplicativos 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

8.73. Vim-9.1.0660

O pacote Vim contém um editor poderoso de texto.

Tempo aproximado de 2,9 UPC

construção:

Espaço em disco exigido: 245 MB



Alternativas ao Vim

Se você preferir outro editor—como o Emacs, Joe ou Nano—por favor, consulte <https://www.linuxfromscratch.org/blfs/view/12.2/postlfs/editors.html> para instruções sugeridas de instalação.

8.73.1. 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) usuário(a) `tester` possa escrever na árvore do fonte:

```
chown -R tester .
```

Agora execute os testes como usuário(a) `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 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 bem sucedido resultará nas palavras `ALL DONE` no arquivo de registro ao completar.

Instale o pacote:

```
make install
```

Muitos(as) usuários(as) reflexivamente digitam `vi` em vez de `vim`. Para permitir a execução do `vim` quando usuários(as) habitualmente digitarem `vi`, crie um link 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-9.1.0660`, tornando-a consistente com o local da documentação para outros pacotes:

```
ln -sv ../vim/vim91/doc /usr/share/doc/vim-9.1.0660
```

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/12.2/postlfs/vim.html>.

8.73.2. Configurando o Vim

Por padrão, **vim** executa em modo incompatível com **vi**. Isso possivelmente seja novo para usuários(as) que tenham usado outros editores no passado. A configuração “*nocompatible*” está incluída abaixo para destacar 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 `.spl` e, opcionalmente, os `.sug` para seu idioma e codificação de carácter a partir de `runtime/spell` para `/usr/share/vim/vim91/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`.

8.73.3. Conteúdo do Vim

Aplicativos 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

8.74. MarkupSafe-2.1.5

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: 508 KB

8.74.1. 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 --no-user --find-links dist Markupsafe
```

8.74.2. Conteúdo do MarkupSafe

Diretório instalado: /usr/lib/python3.12/site-packages/MarkupSafe-2.1.5.dist-info

8.75. Jinja2-3.1.4

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: 2,5 MB

8.75.1. 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 --no-user --find-links dist Jinja2
```

8.75.2. Conteúdo do Jinja2

Diretório instalado: /usr/lib/python3.12/site-packages/Jinja2-3.1.4.dist-info

8.76. Udev originário de Systemd-256.4

O pacote "Udev" contém aplicativos para criação dinâmica de nós de dispositivos.

Tempo aproximado de 0,2 UPC

construção:

Espaço em disco exigido: 144 MB

8.76.1. Instalação do "Udev"

O "Udev" é parte do pacote systemd-256.4. Use o arquivo systemd-256.4.tar.xz como o tarball fonte.

Remova dois grupos desnecessários, `render` e `sgx`, das regras padrão do "udev":

```
sed -i -e 's/GROUP="render"/GROUP="video"/' \
    -e 's/GROUP="sgx", //' rules.d/50-udev-default.rules.in
```

Remova uma regra do "udev" que exige uma instalação completa do "Systemd":

```
sed '/systemd-sysctl/s/^/#/' -i rules.d/99-systemd.rules.in
```

Ajuste os caminhos rigidamente codificados para os arquivos de configuração da rede de intercomunicação para a instalação autônoma do "Udev":

```
sed '/NETWORK_DIRS/s/systemd/udev/' -i src/basic/path-lookup.h
```

Prepare o "Udev" para compilação:

```
mkdir -p build
cd      build

meson setup .. \
    --prefix=/usr \
    --buildtype=release \
    -D mode=release \
    -D dev-kvm-mode=0660 \
    -D link-udev-shared=false \
    -D logind=false \
    -D vconsole=false
```

O significado das opções "meson":

`--buildtype=release`

Essa chave substitui o tipo de construção padrão ("debug"), que produz binários não otimizados.

`-D mode=release`

Desabilite alguns recursos considerados experimentais pelo(a) desenvolvedor(a).

`-D dev-kvm-mode=0660`

A regra padrão do "udev" permitiria que todos(as) os(as) usuários(as) acessassem `/dev/kvm`. Os(As) editores(as) a consideram perigosa. Essa opção a substitui.

`-D link-udev-shared=false`

Essa opção evita que o "udev" se vincule à biblioteca interna compartilhada do "systemd", `libsystemd-shared`. Essa biblioteca foi projetada para ser compartilhada por muitos componentes do "Systemd" e é muito exagerada para uma instalação somente do "udev".

`-D logind=false -D vconsole=false`

Essas opções evitam a geração de vários arquivos de regras do "Udev" pertencentes a outros componentes do "Systemd" que nós não instalaremos.

Obtenha a lista dos auxiliares "Udev" enviados e salve-a em uma variável de ambiente (exportá-la não é estritamente necessário, mas facilita construir como um(a) usuário(a) comum ou usar um gerenciador de pacotes):

```
export udev_helpers=$(grep "'name' :" ../src/udev/meson.build | \
    awk '{print $3}' | tr -d ",'" | grep -v 'udevadm')
```

Construa somente os componentes necessários para o "udev":

```
ninja udevadm systemd-hwdb \
$(ninja -n | grep -Eo '(src/(lib)?udev|rules.d|hwdb.d)/[^ ]*') \
$(realpath libudev.so --relative-to .) \
$udev_helpers
```

Instale o pacote:

```
install -vm755 -d {/usr/lib,/etc}/udev/{hwdb.d,rules.d,network}
install -vm755 -d /usr/{lib,share}/pkgconfig
install -vm755 udevadm /usr/bin/
install -vm755 systemd-hwdb /usr/bin/udev-hwdb
ln -svfn ../bin/udevadm /usr/sbin/udev
cp -av libudev.so{,[0-9]} /usr/lib/
install -vm644 ../src/libudev/libudev.h /usr/include/
install -vm644 src/libudev/*.pc /usr/lib/pkgconfig/
install -vm644 src/udev/*.pc /usr/share/pkgconfig/
install -vm644 ../src/udev/udev.conf /etc/udev/
install -vm644 rules.d/* ../rules.d/README /usr/lib/udev/rules.d/
install -vm644 $(find ../rules.d/*.rules \
-not -name '*power-switch*') /usr/lib/udev/rules.d/
install -vm644 hwdb.d/* ../hwdb.d/{*.hwdb,README} /usr/lib/udev/hwdb.d/
install -vm755 $udev_helpers /usr/lib/udev
install -vm644 ../network/99-default.link /usr/lib/udev/network
```

Instale algumas regras personalizadas e arquivos de suporte úteis em um ambiente LFS:

```
tar -xvf ../../udev-lfs-20230818.tar.xz
make -f udev-lfs-20230818/Makefile.lfs install
```

Instale as páginas de manual:

```
tar -xf ../../systemd-man-pages-256.4.tar.xz \
--no-same-owner --strip-components=1 \
-C /usr/share/man --wildcards '*udev*' '*libudev*' \
'*systemd.link.5' \
'*systemd-'{hwdb,udev.service}.8

sed 's|systemd/network|udev/network|' \
/usr/share/man/man5/systemd.link.5 \
> /usr/share/man/man5/udev.link.5

sed 's/systemd(\\|\\|?-)udev\\1/' /usr/share/man/man8/systemd-hwdb.8 \
> /usr/share/man/man8/udev-hwdb.8

sed 's|lib.*udev|sbin/udev|' \
/usr/share/man/man8/systemd-udev.service.8 \
> /usr/share/man/man8/udev.8

rm /usr/share/man/man*/systemd*
```

Finalmente, desconfigure a variável "udev_helpers":

```
unset udev_helpers
```

8.76.2. Configurando o "Udev"

Informações acerca de dispositivos de hardware são mantidas nos diretórios `/etc/udev/hwdb.d` e `/usr/lib/udev/hwdb.d`. O Udev precisa que essas informações sejam compiladas em uma base de dados binária `/etc/udev/hwdb.bin`. Crie a base de dados inicial:

```
udev-hwdb update
```

Esse comando precisa ser executado sempre que as informações de hardware forem atualizadas.

8.76.3. Conteúdo do "Udev"

Aplicativos instalados:	udevadm, udevd (link simbólico para udevadm) e udev-hwdb
Bibliotecas instaladas:	libudev.so
Diretórios instalados:	/etc/udev e /usr/lib/udev

Descrições Curtas

udevadm	Ferramenta genérica de administração do "udev": controla o processo de segundo plano "udev"; fornece informações a partir da base de dados do "Udev"; monitora "uevents"; aguarda "uevents" finalizarem; testa a configuração do "Udev"; e aciona "uevents" para um dado dispositivo
udev	Um processo de segundo plano que escuta "uevents" no soquete "netlink", cria dispositivos e executa os aplicativos externos configurados em resposta a esses "uevents"
udev-hwdb	Atualiza ou consulta a base de dados de hardware.
libudev	Uma interface de biblioteca para informações de dispositivo do "udev"
/etc/udev	Contém arquivos de configuração do "Udev", permissões de dispositivos e regras para nomeação de dispositivos

8.77. Man-DB-2.12.1

O pacote Man-DB contém aplicativos para encontrar e visualizar páginas de manual.

Tempo aproximado de 0,3 UPC

construção:

Espaço em disco exigido: 43 MB

8.77.1. Instalação do Man-DB

Prepare Man-DB para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/man-db-2.12.1 \
            --sysconfdir=/etc \
            --disable-setuid \
            --enable-cache-owner=bin \
            --with-browser=/usr/bin/lynx \
            --with-vgrind=/usr/bin/vgrind \
            --with-grap=/usr/bin/grap \
            --with-systemdtmpfilesdir= \
            --with-systemdsystemunitdir=
```

O significado das opções do configure:

--disable-setuid

Isso desabilita tornar o aplicativo **man** setuid para o(a) usuário(a) **man**.

--enable-cache-owner=bin

Isso muda a propriedade dos arquivos de cache de abrangência ao sistema para o(a) usuário(a) **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.

--with-systemd...

Esses parâmetros impedem a instalação de diretórios e arquivos desnecessários do systemd.

Compile o pacote:

```
make
```

Para testar os resultados, emita:

```
make check
```

Instale o pacote:

```
make install
```

8.77.2. 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/<ll>` 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

Idioma (código)	Codificação	Idioma (código)	Codificação
Alemão (de)	ISO-8859-1	Húngaro (hu)	ISO-8859-2
Inglês (en)	ISO-8859-1	Japonês (ja)	EUC-JP
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.

8.77.3. Conteúdo do Man-DB

Aplicativos 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-2.12.1

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) usuário(a)
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

8.78. Procps-ng-4.0.4

O pacote Procps-ng contém aplicativos para monitorar processos.

Tempo aproximado de construção: 0,1 UPC

Espaço em disco exigido: 26 MB

8.78.1. Instalação do Procps-ng

Prepare procps-ng para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/procps-ng-4.0.4 \
            --disable-static \
            --disable-kill
```

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.

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
```

8.78.2. Conteúdo do Procps-ng

Aplicativos 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-4.0.4

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 está executando, quantos(as) usuários(as) 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 usuários(as) 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) usuário(a) observe a mudança de saída gerada ao longo do tempo
<code>libproc-2</code>	Contém as funções usadas pela maioria dos aplicativos nesse pacote

8.79. Util-linux-2.40.2

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: 315 MB

8.79.1. 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 \
            --without-systemd \
            --without-systemdsystemunitdir \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.2
```

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(a) não-root:



Atenção

Executar a suíte de teste como o(a) usuário(a) `root` pode ser danoso para o seu 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"
```

Os testes de `hardlink` falharão se o núcleo do anfitrião não tiver a opção `CONFIG_CRYPT_USER_API_HASH` habilitada ou não tiver quaisquer opções fornecendo uma implementação SHA256 (por exemplo, `CONFIG_CRYPT_SHA256`; ou `CONFIG_CRYPT_SHA256_SSSE3`, se a CPU suportar Suplemento SSE3) habilitada. Além disso, o teste `lsfd: inotify` falhará se a opção de núcleo `CONFIG_NETLINK_DIAG` não estiver habilitada.

Instale o pacote:

```
make install
```

8.79.2. Conteúdo do Util-linux

Aplicativos 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, lsipt, 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, rfcill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, taskset, uclampset, ul, umount, uname26 (link para setarch), unshare, utmpdump, uidd, uiddgen, uiddparse, wall, wdcctl, 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-2.40.2 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 usuários(as) 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
findcore	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 IO e prioridade para um aplicativo
ipcmk	Cria vários recursos IPC
ipcrm	Remove o dado recurso de Inter-Process Communication (IPC)
ipcs	Fornece 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 usuários(as) 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 lsdf
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 usuários(as), 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) usuários(as) podem enviar mensagens para o terminal atual do(a) usuário(a)
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
rfkill	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 <code>root</code> se logar; ele normalmente é invocado por init quando o sistema entra em modo de usuário(a) único(a)
swapon	Habilita dispositivos e arquivos para paginação e troca e lista os dispositivos e arquivos atualmente em uso
swapoff	Desabilita dispositivos e arquivos para paginação e troca
swapon	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 <code>setarch</code>
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) usuário(a)
uidd	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) usuários(as) 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 <code>setarch</code>
zramctl	Um aplicativo para configurar e controlar dispositivos zram (disco ram comprimido)
<code>libblkid</code>	Contém rotinas para identificação de dispositivo e extração de token
<code>libfdisk</code>	Contém rotinas para manipular tabelas de partição
<code>libmount</code>	Contém rotinas para montagem e desmontagem de dispositivo de bloco
<code>libsmartcols</code>	Contém rotinas para auxiliar a saída gerada de tela em forma tabular
<code>libuuid</code>	Contém rotinas para gerar identificadores únicos para objetos que possivelmente sejam acessíveis além do sistema local

8.80. E2fsprogs-1.47.1

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, 0,5 UPC em um SSD

Espaço em disco exigido: 98 MB

8.80.1. 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-uuidd    \
             --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.

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
```

8.80.2. 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 suas 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, [então] 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.

8.80.3. Conteúdo do E2fsprogs

Aplicativos instalados: badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2mmpstatus, e2scrub, e2scrub_all, e2undo, e4crypt, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mklost+found, resize2fs e tune2fs

Bibliotecas instaladas: libcom_err.so, libe2p.so, libext2fs.so e libss.so

Diretórios instalados: /usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et e /usr/share/ss

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 aplicativo 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 <code>ext4</code> 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 <code>libss</code>
mke2fs	Cria um sistema de arquivos <code>ext{234}</code> no dispositivo dado
mkfs.ext2	Por padrão cria sistemas de arquivos <code>ext2</code> e é um link rígido para mke2fs
mkfs.ext3	Por padrão cria sistemas de arquivos <code>ext3</code> e é um link rígido para mke2fs
mkfs.ext4	Por padrão cria sistemas de arquivos <code>ext4</code> e é um link rígido para mke2fs
mklost+found	Cria um diretório <code>lost+found</code> em um sistema de arquivos <code>ext{234}</code> ; 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 <code>ext{234}</code>
tune2fs	Ajusta parâmetros ajustáveis do sistema de arquivos em sistema de arquivos <code>ext{234}</code>
<code>libcom_err</code>	A rotina comum de exibição de erro
<code>libe2p</code>	Usado por dumpe2fs , chattr e lsattr
<code>libext2fs</code>	Contém rotinas para habilitar aplicativos de nível de usuário(a) para manipular sistemas de arquivos <code>ext{234}</code>
<code>libss</code>	Usado por debugfs

8.81. Sysklogd-2.6.1

O pacote Sysklogd contém aplicativos para registrar mensagens do sistema, tais como aquelas emitidas pelo núcleo quando coisas incomuns acontecem.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 3,9 MB

8.81.1. Instalação do Sysklogd

Prepare o pacote para compilação:

```
./configure --prefix=/usr      \  
            --sysconfdir=/etc  \  
            --runstatedir=/run \  
            --without-logger
```

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

8.81.2. Configurando Sysklogd

Crie um novo arquivo `/etc/syslog.conf` executando o seguinte:

```
cat > /etc/syslog.conf << "EOF"  
# Inicia /etc/syslog.conf  
  
auth,authpriv.* -/var/log/auth.log  
*.*/auth,authpriv.none -/var/log/sys.log  
daemon.* -/var/log/daemon.log  
kern.* -/var/log/kern.log  
mail.* -/var/log/mail.log  
user.* -/var/log/user.log  
*.emerg *  
  
# Não abre quaisquer portas de internet.  
secure_mode 2  
  
# Termina /etc/syslog.conf  
EOF
```

8.81.3. Conteúdo do Sysklogd

Aplicativo instalado: syslogd

Descrições Curtas

syslogd Registra as mensagens que aplicativos do sistema oferecem para registro [Cada mensagem registrada contém pelo menos um carimbo de data e um nome de dispositivo e normalmente o nome do aplicativo também, porém isso depende do quão confiável o processo de segundo plano de registro é dito ser].

8.82. SysVinit-3.10

O pacote SysVinit contém programas para controlar a inicialização, execução e desligamento do sistema.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 2,8 MB

8.82.1. Instalação do SysVinit

Primeiro, aplique um remendo que remove vários aplicativos instalados por outros pacotes, esclarece uma mensagem e corrige um aviso de compilador:

```
patch -Np1 -i ../sysvinit-3.10-consolidated-1.patch
```

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

8.82.2. Conteúdo do SysVinit

Aplicativos instalados: bootlogd, fstab-decode, halt, init, killall5, poweroff (link para halt), reboot (link para halt), runlevel, shutdown e telinit (link para init)

Descrições Curtas

bootlogd	Registra mensagens de inicialização em um arquivo de registro
fstab-decode	Executa um comando com argumentos codificados para fstab
halt	Normalmente invoca shutdown com a opção <code>-h</code> , porém quando já em nível de execução 0, diz ao núcleo para parar o sistema; ele anota no arquivo <code>/var/log/wtmp</code> que o sistema está sendo desligado
init	O primeiro processo a ser iniciado quando o núcleo tenha inicializado o hardware; ele assume o processo de inicialização e inicia todos os processos especificados no arquivo de configuração dele
killall5	Envia um sinal para todos os processos, exceto os processos na própria sessão dele; ele não matará o shell ancestral dele
poweroff	Diz ao núcleo para parar o sistema e desligar o computador (veja-se halt)
reboot	Diz ao núcleo para reinicializar o sistema (veja-se halt)
runlevel	Informa o nível de execução anterior e o atual, conforme anotado no registro mais recente de nível de execução em <code>/run/utmp</code>
shutdown	Desliga o sistema de uma maneira segura, sinalizando todos os processos e notificando todos(as) os(as) usuários(as) logados(as)
telinit	Informa ao init para qual nível de execução mudar

8.83. Acerca dos Símbolos de Depuração

A maioria dos aplicativos e bibliotecas é, por padrão, compilado com símbolos de depuração inclusos (com a opção `-g` do `gcc`). Isso significa que quando depurar um aplicativo 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 aplicativo 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) usuários(as) 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 aplicativos e bibliotecas.

8.84. Despojando

Esta seção é opcional. Se o(a) pretendo(a) usuário(a) não for um(a) programador(a) e não planeja fazer qualquer depuração do software do sistema, [então] 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 símbolo, de binários e de bibliotecas. Isso não causa nenhum inconveniente real para um(a) usuário(a) 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. Também remove todas as entradas da tabela de símbolo não necessitadas pelo vinculador (para bibliotecas estáticas) ou pelo vinculador dinâmico (para binários vinculados dinamicamente e bibliotecas compartilhadas).

Os símbolos de depuração oriundos de bibliotecas selecionadas são comprimidos com Zlib 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 binário ou arquivo 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, [então] o binário ou biblioteca sendo despojado pode ser destruído; isso pode tornar o sistema completamente inutilizável. 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), [então] 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 inutilizável.**

```
save_usrlib="$(cd /usr/lib; ls ld-linux*[^g])
    libc.so.6
    libthread_db.so.1
    libquadmath.so.0.0.0
    libstdc++.so.6.0.33
    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=zlib $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-2.43.1.so
    libsframe.so.1.0.0
    libhistory.so.8.2
    libncursesw.so.6.5
    libm.so.6
    libreadline.so.8.2
    libz.so.1.3.1
    libzstd.so.1.5.6
    $(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. Limpando

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) usuário(a) 'tester' criada no início do capítulo anterior.

```
userdel -r tester
```

Capítulo 9. Configuração do Sistema

9.1. Introdução

Inicializar um sistema Linux envolve várias tarefas. O processo precisa montar ambos sistemas de arquivos virtual e real, inicializar dispositivos, verificar sistemas de arquivos para integridade, montar e ativar quaisquer partições ou arquivos de troca, configurar o relógio do sistema, ativar a rede de comunicação, iniciar quaisquer processos de segundo plano exigidos pelo sistema e realizar quaisquer outras tarefas personalizadas especificadas pelo(a) usuário(a). Esse processo precisa estar organizado para garantir que as tarefas sejam realizadas na ordem correta e executadas o mais rápido possível.

9.1.1. System V

System V é o processo clássico de inicialização que tem sido usado em sistemas Unix e semelhantes a Unix, tais como o Linux, desde cerca de 1983. Ele consiste de um aplicativo pequeno, **init**, que configura processos básicos, tais como **login** (via **getty**), e executa um script. Esse script, usualmente chamado de **rc**, controla a execução de um conjunto de scripts adicionais que realizam as tarefas exigidas para inicializar o sistema.

O aplicativo **init** é controlado pelo arquivo `/etc/inittab` e está organizado em níveis de execução que podem ser escolhidos pelo(a) usuário(a). No LFS, eles são usados como segue:

```
0 — parar
1 — Modo de usuário(a) único(a)
2 — Definível pelo(a) usuário(a)
3 — Modo de multi usuário(a) completo
4 — Definível pelo(a) usuário(a)
5 — Modo de multi usuário(a) completo com gerenciador de tela
6 — reinicializar
```

O nível de execução padrão usual é 3 ou 5.

Vantagens

- Sistema estabelecido, bem compreendido.
- Fácil de personalizar.

Desvantagens

- Possivelmente seja mais lento inicializar. Um sistema LFS básico de velocidade média toma de 8 a 12 segundos, onde o tempo de inicialização é medido desde a primeira mensagem do núcleo até o prompt de login. A conectividade da rede de comunicação tipicamente é estabelecida cerca de 2 segundos depois do prompt de login.
- Processamento em série de tarefas de inicialização. Isso está relacionado ao ponto anterior. Um atraso em qualquer processo, tal como uma verificação de sistema de arquivos, atrasará o processo inteiro de inicialização.
- Não suporta diretamente recursos avançados, como grupos de controle (**cgroups**) e agendamento de compartilhamento justo por usuário(a).
- Adicionar scripts exige decisões de sequenciamento estático, manuais.

9.2. LFS-Bootscripts-20240825

O pacote LFS-Bootscripts contém um conjunto de scripts para iniciar/parar o sistema LFS na inicialização/desligamento. Os arquivos e procedimentos de configuração necessários para personalizar o processo de inicialização estão descritos nas seções seguintes.

Tempo aproximado de construção: menos que 0,1 UPC

Espaço em disco exigido: 248 KB

9.2.1. Instalação do LFS-Bootscripts

Instale o pacote:

```
make install
```

9.2.2. Conteúdo do LFS-Bootscripts

Scripts instalados: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev e udev_retry

Diretórios instalados: /etc/rc.d, /etc/init.d (link simbólico), /etc/sysconfig, /lib/services e /lib/lsb (link simbólico)

Descrições Curtas

checkfs Verifica a integridade dos sistemas de arquivos antes que eles sejam montados (com a exceção dos sistemas de arquivos baseados em diário e em rede de comunicação)

cleanfs Remove os arquivos que não deveriam ser preservados entre as reinicializações, tais como aqueles em `/run/` e `/var/lock/`; ele recria `/run/utmp` e remove os arquivos possivelmente presentes `/etc/nologin`, `/fastboot` e `/forcefsck`

console Carrega a tabela correta de mapa de tecla para o esquema de teclado desejado; ele também configura a fonte de tela

functions Contém funções comuns, tais como de verificação de erro e de situação, que são usadas por vários scripts de inicialização

halt Pára o sistema

ifdown Pára um dispositivo de rede de comunicação

ifup Inicializa um dispositivo de rede de comunicação

localnet Configura o nome de dispositivo do sistema e dispositivo local de loopback

modules Carrega módulos do núcleo listados em `/etc/sysconfig/modules`, usando argumentos que também são dados lá

mountfs Monta todos os sistemas de arquivos, exceto aqueles que estejam marcados como *noauto* ou são baseados em rede de comunicação

mountvirtfs Monta os sistemas de arquivos virtuais do núcleo, tais como o `proc`

network Configura as interfaces da rede de comunicação, tais como placas de rede de comunicação, e configura o gateway padrão (onde aplicável)

rc O script mestre de controle de nível de execução; ele é responsável por executar todos os outros scripts de inicialização, um por um, em uma sequência determinada pelos nomes dos links simbólicos para aqueles outros scripts de inicialização

reboot	Reinicializa o sistema
sendsignals	Garante que cada processo seja terminado antes que o sistema reinicialize ou pare
setclock	Reconfigura o relógio do sistema para hora local se o relógio do hardware não estiver configurado para UTC
ipv4-static	Fornecer a funcionalidade necessária para atribuir um endereço estático de Internet Protocol (IP) para uma interface de rede de comunicação
swap	Habilita e desabilita arquivos e partições de troca
sysctl	Carrega valores de configuração do sistema a partir do <code>/etc/sysctl.conf</code> , se esse arquivo existir, para dentro do núcleo em execução
sysklogd	Inicia e pára os processos de segundo plano de registro do sistema e do núcleo
template	Um modelo para criar scripts de inicialização personalizados para outros processos de segundo plano
udev	Prepara o diretório <code>/dev</code> e inicia o processo de segundo plano Udev
udev_retry	Tenta novamente uevents do Udev e copia arquivos gerados de regras de <code>/run/udev</code> para <code>/etc/udev/rules.d</code> se exigido

9.3. Visão Geral do Manuseio de Dispositivo e de Módulo

No Capítulo 8, nós instalamos o processo de segundo plano Udev quando udev 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 usuário(a). Com essa representação visível ao espaço de usuário(a), tornou-se possível desenvolver um substituto de espaço de usuário(a) 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 usuário(a) 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 usuário(a) 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 links 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-6.10.5, 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-6.10.5, 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”.

Se o módulo em questão não é um encapsulador e é útil por ele próprio, [então] configure o script de inicialização **modules** para carregar esse módulo na inicialização do sistema. Para fazer isso, adicione o nome do módulo ao arquivo `/etc/sysconfig/modules` em uma linha separada. Isso funciona para módulos encapsuladores também, mas é abaixo do ideal nesse caso.

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 as tuas próprias regras que fazem links simbólicos com nomes estáveis baseados em alguns atributos estáveis do dispositivo, tais como um número de série ou a saída gerada dos vários utilitários `*_id` instalados pelo "Udev". Veja-se a "Seção 9.4, "Gerenciando Dispositivos"" e "Seção 9.5, "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 usuário(a) 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. Dispositivos da Rede de Comunicação

O Udev, por padrão, nomeia dispositivos da rede de comunicação de acordo com dados de Firmware/BIOS ou características físicas, como o barramento, slot ou endereço MAC. O propósito dessa convenção de nomenclatura é o de garantir que dispositivos da rede de comunicação sejam nomeados consistentemente e não baseados em quando a placa de rede de comunicação foi descoberta. Em versões mais antigas do Linux—em um computador com duas placas de rede de comunicação feitas por Intel e Realtek, por exemplo—a placa de rede de comunicação fabricada pela Intel talvez se torne `eth0`, enquanto que e a placa Realtek se tornou `eth1`. Depois de uma reinicialização, as placas poderiam, às vezes, serem renumeradas da maneira inversa.

No novo esquema de nomenclatura, nomes típicos de dispositivo da rede de comunicação são alguma coisa como `enp5s0` ou `wlp3s0`. Se essa convenção de nomenclatura não for desejada, [então] o esquema de nomenclatura tradicional, ou um esquema personalizado, pode ser implementado.

9.4.1.1. Desabilitando Nomenclatura Persistente na Linha de Comando do Núcleo

O esquema tradicional de nomenclatura usando `eth0`, `eth1`, etc., pode ser restaurado adicionando-se `net.ifnames=0` na linha de comando do núcleo. Isso é mais apropriado para sistemas que tenham apenas um dispositivo ethernet de um tipo específico. Laptops frequentemente tem duas conexões ethernet nomeadas de `eth0` e `wlan0`; tais laptops também conseguem usar esse método. A linha de comando está no arquivo de configuração do GRUB. Veja-se Seção 10.4.4, "Criando o Arquivo de Configuração do GRUB."

9.4.1.2. Criando Regras Personalizadas do Udev

O esquema de nomenclatura pode ser personalizado criando-se regras personalizadas do Udev. Um script foi incluído que gera as regras iniciais. Gere essas regras executando:

```
bash /usr/lib/udev/init-net-rules.sh
```

Agora, inspecione o arquivo `/etc/udev/rules.d/70-persistent-net.rules`, para descobrir qual nome foi atribuído a qual dispositivo da rede de comunicação:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```



Nota

Em alguns casos, tais como quando endereços MAC tenham sido atribuídos para uma placa de rede de comunicação manualmente, ou em um ambiente virtual, como Qemu ou Xen, o arquivo das regras da rede de comunicação possivelmente não seja gerado, pois endereços não são atribuídos consistentemente. Nesses casos, esse método não pode ser usado.

O arquivo começa com um bloco de comentário seguido por duas linhas para cada NIC. A primeira linha para cada NIC é uma descrição comentada mostrando os IDs de hardware delas (por exemplo, fornecedor(a) de PCI delas e IDs de dispositivo, se ela for uma placa PCI), juntamente com o controlador delas (entre parênteses, se o controlador puder ser encontrado). Nem o ID de hardware, nem o controlador, é usado para determinar quais nomes dar para uma interface; essa informação é somente para referência. A segunda linha é a regra do Udev que corresponde a essa NIC e atualmente atribui a ela um nome.

Todas as regras do Udev são compostas de várias palavras chave, separadas por vírgulas e espaços em branco opcionais. Aqui estão as palavras chave e uma explicação de cada uma:

- `SUBSYSTEM=="net"` - Isso diz ao Udev para ignorar dispositivos que não sejam placas da rede de comunicação.
- `ACTION=="add"` - Isso diz ao Udev para ignorar essa regra para um uevent que não seja um adicionar (uevents "remove" e "mudar" também acontecem, porém não precisam renomear interfaces da rede de comunicação).
- `DRIVERS=="?*"` - Isso existe, de forma que o Udev ignorará sub-interfaces VLAN ou bridge (pois essas sub-interfaces não tem controladores). Essas sub-interfaces são puladas, pois o nome que seria atribuído conflitaria com os dispositivos ancestrais delas.
- `ATTR{address}` - O valor dessa palavra chave é o endereço MAC da NIC.
- `ATTR{type}=="1"` - Isso garante que a regra corresponda somente à interface primária no caso de certos controladores sem fios os quais criam múltiplas interfaces virtuais. As interfaces secundárias são puladas pela mesma razão que sub-interfaces VLAN e bridge são puladas: existiria um conflito de nome do contrário.
- `NAME` - O valor dessa palavra chave é o nome que o Udev atribuirá para essa interface.

O valor de `NAME` é a parte importante. Assegure-se de que você sabe qual nome foi atribuído para cada uma das suas placas da rede de comunicação antes de prosseguir, e tenha certeza de usar esse valor `NAME` quando criar seus arquivos de configuração da rede de comunicação.

Mesmo se o arquivo de regras personalizadas do "udev" for criado, o "udev" possivelmente ainda atribuirá um ou mais nomes alternativos para uma "NIC" baseados nas características físicas. Se uma regra personalizada do "udev" renomeasse alguma "NIC" usando um nome já atribuído como um nome alternativo de outra "NIC", [então] essa regra do "udev" falharia. Se esse problema ocorrer, [então] você poderá criar o arquivo de configuração `/etc/udev/network/99-default.link` com uma política de atribuição alternativa vazia, substituindo o arquivo de configuração padrão `/usr/lib/udev/network/99-default.link`:

```
sed -e '/^AlternativeNamesPolicy/s/=.*/=/' \
     /usr/lib/udev/network/99-default.link \
     > /etc/udev/network/99-default.link
```

9.4.2. Links Simbólicos de CD-ROM

Alguns aplicativos que você possivelmente queira instalar posteriormente (por exemplo, vários reprodutores de mídia) esperam que os links simbólicos `/dev/cdrom` e `/dev/dvd` existam, e apontem para um dispositivo de CD-ROM ou de DVD-ROM. Também, possivelmente seja conveniente colocar referências a esses links simbólicos em

`/etc/fstab`. O Udev vem com um script que gerará arquivos de regras para criar esses links simbólicos para você, dependendo dos recursos de cada dispositivo, mas você precisa decidir qual dos dois modos de operação você deseja ter para o script usar.

Primeiro, o script pode operar em modo “por-caminho” (usado por padrão para dispositivos USB e FireWire), onde as regras que ele cria dependem do caminho físico para o dispositivo de CD ou de DVD. Segundo, ele pode operar em modo “por-id” (padrão para dispositivos IDE e SCSI), onde as regras que ele cria dependem das sequências de caracteres de identificação armazenadas no próprio dispositivo de CD ou de DVD. O caminho é determinado pelo script `path_id` do Udev, e as sequências de caracteres de identificação são lidas a partir do hardware pelos aplicativos `ata_id` ou `scsi_id` dele, dependendo de qual tipo de dispositivo você tenha.

Existem vantagens para cada abordagem; a abordagem correta depende de que tipos de mudanças de dispositivo possivelmente aconteçam. Se você espera o caminho físico para o dispositivo (isto é, as portas e (ou) slots aos quais ele se pluga) mudar, por exemplo porque você planeja mover a unidade para uma porta IDE diferente ou um conector USB diferente, então você deveria usar o modo “por-id”. Por outro lado, se você espera que a identificação do dispositivo mude, por exemplo porque ele possivelmente morra, e você pretende substituí-lo por um dispositivo diferente que pluga nos mesmos conectores, então você deveria usar o modo “por-caminho”.

Se ambos os tipos de mudanças são possíveis com a sua unidade, então escolha um modo baseado no tipo de mudança que você espera acontecer mais frequentemente.



Importante

Dispositivos externos (por exemplo, uma unidade de CD conectada via USB) não deveria usar persistência por caminho, porque cada vez que o dispositivo for plugado em uma nova porta externa, o caminho físico dele mudará. Todos os dispositivos conectados externamente terão esse problema se você escrever regras do Udev para reconhecê-los pelo caminho físico deles; o problema não está limitado a unidades de CD e de DVD.

Se você deseja ver os valores que os scripts do Udev usarão, então para o dispositivo apropriado de CD-ROM, encontre o diretório correspondente sob `/sys` (por exemplo, isso pode ser `/sys/block/hdd`) e execute um comando similar ao seguinte:

```
udevadm test /sys/block/hdd
```

Olhe para as linhas contendo a saída gerada de vários aplicativos `*_id`. O modo “por-id” usará o valor `ID_SERIAL` se ele existir e não estiver vazio; do contrário ele usará uma combinação de `ID_MODEL` e `ID_REVISION`. O modo “por-caminho” usará o valor `ID_PATH`.

Se o modo padrão não for adequado para a sua situação, então a seguinte modificação pode ser feita para o arquivo `/etc/udev/rules.d/83-cdrom-symlinks.rules`, como se segue (onde `mode` é um de “por-id” ou “por-caminho”):

```
sed -e 's/"write_cd_rules"/"write_cd_rules mode"/' \  
-i /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Observe que não é necessário criar os arquivos de regras ou links simbólicos neste momento, porque você montou vinculadamente o diretório do sistema anfitrião `/dev` dentro do sistema LFS, e nós assumimos que os links simbólicos existem no anfitrião. As regras e links simbólicos serão criados na primeira vez que você inicializar seu sistema LFS.

Entretanto, se você tiver múltiplos dispositivos de CD-ROM, então os links simbólicos gerados naquele momento possivelmente apontem para dispositivos diferentes dos que eles apontam em seu anfitrião, porque os dispositivos não são descobertos em uma ordem previsível. As atribuições criadas quando você inicializar o sistema LFS pela primeira vez serão estáveis, de forma que isso é um problema somente se você precisar dos links simbólicos em ambos os sistemas para apontarem para o mesmo dispositivo. Se você precisar disso, então inspecione (e possivelmente edite) o arquivo `/etc/udev/rules.d/70-persistent-cd.rules` gerado após a inicialização, para ter certeza que os links simbólicos atribuídos correspondem às suas necessidades.

9.4.3. 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 9.5, “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+="tvtuner"

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. Configuração Geral da Rede de Comunicação

9.5.1. Criando Arquivos de Configuração de Interface de Rede de Comunicação

Os arquivos em `/etc/sysconfig/` usualmente determinam quaisquer interfaces são levantadas e derrubadas pelo script da rede de comunicação. Esse diretório deveria conter um arquivo para cada interface a ser configurada, tal como `ifconfig.xyz`, onde “xyz” descreve a placa da rede de comunicação. O nome da interface (por exemplo, `eth0`) usualmente é apropriado. Cada arquivo contém os atributos de uma interface, tais como endereço(s) IP dela, máscaras de sub-rede, e por aí vai. A base do nome do arquivo precisa ser *ifconfig*.



Nota

Se o procedimento na seção anterior não foi usado, [então] o Udev atribuirá nomes de interface da placa de rede de comunicação baseados em características físicas do sistema, tais como `enp2s1`. Se você não tem certeza qual é seu nome de interface, [então] você sempre pode executar `ip link` ou `ls /sys/class/net` depois que você tenha inicializado o seu sistema.

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 (instalado na Seção 8.76, “Udev originário de Systemd-256.4”) não executará até que o sistema LFS seja inicializado. Assim, os nomes de interface no sistema LFS não pode sempre ser determinado executando-se aqueles comandos na distribuição anfitriã, *mesmo no ambiente chroot*.

O seguinte comando cria um arquivo modelo para o dispositivo *eth0* com um endereço estático de IP:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Os valores em *itálico* precisam ser mudados em cada arquivo para configurar corretamente.

Se a variável "ONBOOT" estiver configurada para "yes", [então] o conjunto de comandos sequenciais da rede de comunicação do "System V" levantará a "Network Interface Card" ("NIC") durante o processo de inicialização do sistema. Se configurado para qualquer coisa exceto "yes", [então] a "NIC" será ignorada pelo conjunto de comandos sequenciais da rede de comunicação e não será iniciada automaticamente. As interfaces podem ser manualmente iniciadas ou paradas com os comandos "ifup" e "ifdown".

A variável IFACE define o nome da interface, por exemplo, eth0. Ela é exigida para todos os arquivos de configuração do dispositivo da rede de comunicação. A extensão do nome do arquivo precisa corresponder a esse valor.

A variável SERVICE define o método usado para obter o endereço de IP. O pacote LFS-Bootscripsts tem um formato modular de atribuição de IP, e criar arquivos adicionais no diretório /lib/services/ permite outros métodos de atribuição de IP. Isso é comumente usado para Dynamic Host Configuration Protocol (DHCP), o qual é endereçado no livro BLFS.

A variável GATEWAY deveria conter o endereço padrão de IP do gateway, se um estiver presente. Se não, então comente a variável inteiramente.

A variável PREFIX especifica o número de bits usados na sub-rede. Cada segmento de um endereço de IP é 8 bits. Se a máscara de rede da sub-rede for 255.255.255.0, então ela está usando os primeiros três segmentos (24 bits) para especificar o número da rede de comunicação. Se a máscara de rede for 255.255.255.240, [então] a sub-rede está usando os primeiros 28 bits. Prefixos mais longos que 24 bits são comumente usados por DSL e Internet Service Providers (ISPs) baseados em cabos. Nesse exemplo (PREFIX=24), a máscara de rede é 255.255.255.0. Ajuste a variável PREFIX de acordo com sua sub-rede específica. Se omitida, [então] o PREFIX padrão é 24.

Para mais informação veja-se a página de manual do ifup.

9.5.2. Criando o Arquivo /etc/resolv.conf

O sistema precisará de alguma meio de obter 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 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. Crie o arquivo executando o seguinte:

```
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 IP do servidor de nome>* pelo endereço de IP do DNS mais apropriado para a configuração. Frequentemente existirá mais que uma entrada (exigências demandam servidores secundários para recurso de substituto). Se você precisa ou quer somente um servidor de DNS, [então] remova a segunda linha *servidornome* do arquivo. O endereço de IP também possivelmente seja um roteador na rede local de comunicação.



Nota

Os endereços do DNS do Google Public IPv4 são 8.8.8.8 e 8.8.4.4.

9.5.3. 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 "<ifs>" > /etc/hostname
```

`<ifs>` precisa ser substituído pelo nome dado para o computador. Não informe o Fully Qualified Domain Name (FQDN) aqui. Essa informação vai no arquivo `/etc/hosts`.

9.5.4. 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) usuários(as) não tem isso), assegure-se de que o endereço de IP está no intervalo de endereço privado de IP da rede de comunicação. 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` executando:

```
cat > /etc/hosts << "EOF"
# Início do /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <NOMEHOST>
<192.168.1.1> <FQDN> <NOMEHOST> [apelido1] [apelido2 ...]
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters

# Fim do /etc/hosts
EOF
```

Os valores `<192.168.1.1>`, `<FQDN>` e `<NOMEHOST>` precisam ser mudados para usuários(as) ou exigências específicos(as) (se atribuído um endereço de IP por um(a) administrador(a) da rede de comunicação/sistema e a máquina estará conectada a uma rede existente de comunicação). O(s) nome(s) de apelido(s) opcional(is) pode(m) ser omitido(s).

9.6. Uso e Configuração do Script de Inicialização do System V

9.6.1. Como os Scripts de Inicialização do System V funcionam?

Esta versão do LFS usa um aparato especial de inicialização chamado SysVinit, baseado em uma série de *níveis de execução*. O procedimento de inicialização pode ser bem diferente de um sistema para outro; o fato de que as coisas funcionam de uma maneira em uma distribuição específica do Linux não garante que elas funcionarão da mesma forma no LFS. O LFS tem a própria maneira de fazer as coisas, mas ele respeita os padrões geralmente aceitos.

Existe um procedimento alternativo de inicialização chamado **systemd**. Nós não mais discutiremos esse processo de inicialização aqui. Para uma descrição detalhada, visite-se <https://www.linux.com/training-tutorials/understanding-and-using-systemd/>.

O "SysVinit" (o qual será referido como "init" daqui pra frente) usa um esquema de níveis de execução. Existem sete níveis de execução, numerados de 0 a 6. (Atualmente, existem mais níveis de execução, mas os outros são para casos especiais e geralmente não são usados. Veja-se "*init(8)*" para mais detalhes). Cada um dos sete corresponde às ações que o computador é suposto realizar quando ele inicia ou desliga. O nível de execução padrão é o 3. Aqui estão as descrições dos diferentes níveis de execução conforme eles estão implementados no LFS:

- 0: parar o computador
- 1: Modo de usuário(a) único(a)
- 2: Reservado para personalização, do contrário faz o mesmo que 3
- 3: Modo de multi-usuário(a), com rede de comunicação
- 4: Reservado para personalização, do contrário faz o mesmo que 3
- 5: Mesmo que 4, ele é usado usualmente para login GUI (como o **gdm** do GNOME ou o **lxdm** do LXDE)
- 6: reinicializar o computador



Nota

Classicamente, o nível de execução 2 acima era definido como “modo de multiusuário(a), sem rede de comunicação”, porém isso somente foi o caso muitos anos atrás quando múltiplos(as) usuários(as) podiam se conectar a um sistema via portas seriais. No ambiente da atualidade isso não faz sentido e agora nós dizemos que ele está “reservado”.

9.6.2. Configurando o SysVinit

Durante a inicialização do núcleo, o primeiro aplicativo que é executado (se não substituído na linha de comando) é o **init**. Esse aplicativo lê o arquivo de inicialização `/etc/inittab`. Crie esse arquivo com:

```
cat > /etc/inittab << "EOF"
# Início do /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc S

10:0:wait:/etc/rc.d/init.d/rc 0
11:S1:wait:/etc/rc.d/init.d/rc 1
12:2:wait:/etc/rc.d/init.d/rc 2
13:3:wait:/etc/rc.d/init.d/rc 3
14:4:wait:/etc/rc.d/init.d/rc 4
15:5:wait:/etc/rc.d/init.d/rc 5
16:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S06:once:/sbin/sulogin
sl:1:respawn:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# Fim do /etc/inittab
EOF
```

Uma explicação desse arquivo de inicialização está na página de manual para *inittab*. No LFS, o comando chave é **rc**. O arquivo de inicialização acima instrui **rc** a executar todos os scripts começando com um **S** no diretório `/etc/rc.d/rcS.d` seguido por todos os scripts começando com um **S** no diretório `/etc/rc.d/rc?.d` onde o ponto de interrogação é especificado pelo valor de `initdefault`.

Como uma conveniência, o script **rc** lê uma biblioteca de funções em `/lib/lsb/init-functions`. Essa biblioteca também lê um arquivo de configuração opcional, `/etc/sysconfig/rc.site`. Quaisquer dos parâmetros de configuração do sistema descritos em seções subsequentes podem ser colocados nesse arquivo, permitindo a consolidação de todos os parâmetros do sistema nesse único arquivo.

Como uma conveniência de depuração, o script de funções também registra todas as saídas geradas em `/run/var/bootlog`. Dado que o diretório `/run` é um tmpfs, esse arquivo não é persistente ao longo de inicializações; entretanto, ele é adicionado ao arquivo mais permanente `/var/log/boot.log` ao final do processo de inicialização.

9.6.2.1. Mudando Níveis de Execução

A mudança de níveis de execução é feita com **init** `<runlevel>`, onde `<runlevel>` é o nível de execução alvo. Por exemplo, para reinicializar o computador, um(a) usuário(a) poderia emitir o comando **init 6**, o qual é um apelido para o comando **reboot**. Da mesma forma, **init 0** é um apelido para o comando **halt**.

Existe um número de diretórios sob `/etc/rc.d` que se parecem com `rc?.d` (onde `?` é o número do nível de execução) e `rcS.d`, todos contendo um número de links simbólicos. Alguns links começam com um **K**; os outros começam com um **S**, e todos eles tem dois números seguindo a letra inicial. O **K** significa parar (kill) um serviço e o **S** significa iniciar um serviço. Os números determinam a ordem na qual os scripts são executados, de 00 a 99—quanto menor o número, mais cedo o script executa. Quando **init** alterna para outro nível de execução, os serviços adequados são tanto iniciados quanto parados, dependendo do nível de execução escolhido.

Os scripts reais estão em `/etc/rc.d/init.d`. Eles fazem o trabalho atual e os links simbólicos todos apontam para eles. Os links K e os links S apontam para o mesmo script em `/etc/rc.d/init.d`. Isso é porque os scripts podem ser chamados com parâmetros diferentes como `start`, `stop`, `restart`, `reload` e `status`. Quando um link K é encontrado, o script apropriado é executado com o argumento `stop`. Quando um link S é encontrado, o script apropriado é executado com o argumento `start`.

Estas são descrições do que os argumentos fazem os scripts fazer:

start

O serviço é iniciado.

stop

O serviço é parado.

restart

O serviço é parado e então iniciado novamente.

reload

A configuração do serviço é atualizada. Isso é usado depois que o arquivo de configuração de um serviço foi modificado, quando o serviço não precisa ser reiniciado.

status

Diz se o serviço está executando e com quais PIDs.

Sinta-se livre para modificar a maneira como o processo de inicialização funciona (afinal de contas, este é seu próprio sistema LFS). Os arquivos dados aqui são um exemplo de como isso pode ser feito.

9.6.3. Scripts de Inicialização do Udev

O script de iniciação `/etc/rc.d/init.d/udev` inicia o **udev**, aciona quaisquer dispositivos "plugue frio" que já tenham sido criados pelo núcleo e aguarda por quaisquer regras para completar. O script também desconfigura o manuseador do uevent do padrão do `/sbin/hotplug`. Isso é feito, pois o núcleo não mais precisa chamar um binário externo. Em vez disso, o **udev** escutará em um soquete de link de rede os uevents que o núcleo gera.

O script `/etc/rc.d/init.d/udev_retry` se ocupa de re-acionar eventos para subsistemas cujas regras possivelmente dependam de sistemas de arquivos que não estão montados até que o script **mountfs** seja executado (em particular, `/usr` e `/var` possivelmente causem isso). Esse script executa depois do script **mountfs**, de forma que aquelas regras (se re-acionadas) deveriam prosperar na segunda vez. Ele é configurado pelo arquivo `/etc/sysconfig/udev_retry`; quaisquer palavras nesse arquivo outras que comentários são consideradas nomes de subsistema para acionar ao tempo de re-tentativa. Para encontrar o subsistema de um dispositivo, use **udevadm info --attribute-walk <dispositivo>**, onde `<dispositivo>` é um caminho absoluto em `/dev` ou `/sys`, tais como `/dev/sr0` ou `/sys/class/rtd`.

Para informação acerca de carregamento de módulo do núcleo e udev, veja-se Seção 9.3.2.3, "Carregamento de Módulo."

9.6.4. Configurando o Relógio do Sistema

O script **setclock** lê a hora a partir do relógio do hardware, também conhecido como relógio do BIOS ou do Complementary Metal Oxide Semiconductor (CMOS). Se o relógio do hardware estiver configurado para UTC, [então] esse script converterá a hora do relógio do hardware para a hora local usando o arquivo `/etc/localtime` (o qual diz ao aplicativo **hwclock** qual fuso horário usar). Não existe maneira de detectar se o relógio do hardware está ou não configurado para UTC, de forma que isso precisa ser configurado manualmente.

O aplicativo **setclock** é executado via udev quando o núcleo detecta o recurso do hardware em consequência da inicialização. Ele também pode ser executado manualmente com o parâmetro `pare` para armazenar a hora do sistema para o relógio CMOS.

Se você não conseguir lembrar se o relógio do hardware está ou não configurado para UTC, [então] descubra executando o comando `hwclock --localtime --show`. Isso mostrará o que é a hora atual 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 número apropriado de horas para o seu 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.

Mude o valor da variável `UTC` abaixo para um valor de `0` (zero) se o relógio do hardware *NÃO* estiver configurado para hora UTC.

Crie um novo arquivo `/etc/sysconfig/clock` executando o seguinte:

```
cat > /etc/sysconfig/clock << "EOF"
# Início do /etc/sysconfig/clock

UTC=1

# Configure isto para quaisquer opções que você pudesse precisar dar para hwclock,
# tais como tipo do relógio de hardware de máquina para Alphas.
CLOCKPARAMS=

# Fim do /etc/sysconfig/clock
EOF
```

Uma boa dica explicando como lidar com hora no LFS está disponível em <https://www.linuxfromscratch.org/hints/downloads/files/time.txt>. Ela explica problemas como fusos horários, UTC e a variável de ambiente `TZ`.



Nota

Os parâmetros `CLOCKPARAMS` e `UTC` também possivelmente sejam configurados no arquivo `/etc/sysconfig/rc.site`.

9.6.5. Configurando o Console do Linux

Esta seção discute como configurar o script de inicialização **console** que configura o mapa de teclado, fonte do console e nível de registro do núcleo do console. Se caracteres não-ASCII (por exemplo, o sinal de direitos autorais, o sinal da libra britânica e o símbolo do Euro) não serão usados e o teclado for um dos Estados Unidos da América do Norte, [então] muito desta seção pode ser pulada. Sem o arquivo de configuração, (ou configurações equivalentes em `rc.site`), o script de inicialização **console** não fará nada.

O conjunto de comandos sequenciais "**console**" lê o arquivo `/etc/sysconfig/console` para informação de configuração. Decida qual mapa de teclas e fonte de tela serão usados. Vários "HOWTOs" específicos de idiomas também podem ajudar com isso; veja-se "<https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>". Se ainda em dúvida, [então] olhe nos diretórios `/usr/share/keymaps` e `/usr/share/consolefonts` para mapas de teclas válidos e fontes de tela. Leiam-se as páginas de manual "`loadkeys(1)`" e "`setfont(8)`" para determinar os argumentos corretos para esses aplicativos.

O arquivo `/etc/sysconfig/console` deveria conter linhas da forma: `"VARIÁVEL=valor"`. As seguintes variáveis são reconhecidas:

LOGLEVEL

Essa variável especifica o nível de registro para mensagens do núcleo enviadas para o console, conforme configurado por "`dmesg -n`". Níveis válidos são de "1" (sem mensagens) até "8". O nível padrão é "7", que é bastante verboso.

KEYMAP

Essa variável especifica os argumentos para o aplicativo "`loadkeys`", tipicamente, o nome do mapa de teclas a carregar, por exemplo, `"it"`. Se essa variável não estiver configurada, [então] o conjunto de comandos

sequenciais de inicialização não executará o aplicativo **"loadkeys"** e o mapa padrão de teclas do núcleo será usado. Observe que uns poucos mapas de teclas tem múltiplas versões com o mesmo nome ("cz" e variantes dele em "qwerty/" e "qwertz/"; "es" em "olpc/" e "qwerty/"; e "trf" em "fgGiod/" e "qwerty/"). Nesses casos, o diretório ancestral também deveria ser especificado (por exemplo, "qwerty/es") para garantir que o mapa de teclas adequado seja carregado.

KEYMAP_CORRECTIONS

Essa (raramente usada) variável especifica os argumentos para a segunda chamada ao aplicativo **"loadkeys"**. Isso é útil se o mapa padrão de teclas não for completamente satisfatório e um pequeno ajuste tenha que ser feito. Por exemplo, para incluir o símbolo do Euro em um mapa de teclas que normalmente não o tem, configure essa variável para "euro2".

FONT

Essa variável especifica os argumentos para o aplicativo **"setfont"**. Tipicamente, isso inclui o nome da fonte, "-m", e o nome do mapa de caracteres de aplicação a carregar. Por exemplo, para a finalidade de carregar a fonte "lat1-16" juntamente com o mapa de caracteres de aplicação "8859-1" (apropriado nos Estados Unidos da América do Norte), configure essa variável para "lat1-16 -m 8859-1". Em modo "UTF-8", o núcleo usa o mapa de caracteres de aplicação para converter os códigos de tecla de 8 bits para "UTF-8". Dessa maneira, o argumento do parâmetro "-m" deveria ser configurado para a codificação dos códigos compostos de tecla no mapa de teclas.

UNICODE

Configure essa variável para "1", "yes" ou "true" para a finalidade de colocar o console em modo "UTF-8". Isso é útil em localidades baseadas em "UTF-8" e danoso em outras.

LEGACY_CHARSET

Para muitos esquemas de teclado, não existe mapa padrão de teclado Unicode no pacote Kbd. O script de inicialização **console** converterá um mapa de teclas disponível para UTF-8 em tempo real se essa variável estiver configurada para a codificação do mapa disponível de teclas não-UTF-8.

Alguns exemplos:

- Usaremos `C.UTF-8` como localidade para sessões interativas no console do Linux na Seção 9.7, "Configurando a Localidade do Sistema," de forma que deveríamos configurar `UNICODE` para 1. E as fontes do console enviadas 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 travessão do Unicode em inglês). Portanto, configure uma delas, por exemplo `Lat2-Terminus16.psfu.gz` como fonte padrão do console:

```
cat > /etc/sysconfig/console << "EOF"
# Inicia /etc/sysconfig/console

UNICODE="1"
FONT="Lat2-Terminus16"

# Termina /etc/sysconfig/console
EOF
```

- Para uma configuração não-Unicode, somente as variáveis `KEYMAP` e `FONT` geralmente são necessárias. Por exemplo, para uma configuração em polonês, alguém usaria:

```
cat > /etc/sysconfig/console << "EOF"
# Início do /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# Fim do /etc/sysconfig/console
EOF
```

- Como mencionado acima, às vezes é necessário ajustar um mapa padrão de teclas um pouco. O exemplo seguinte adiciona o símbolo do Euro ao mapa de teclas alemão:

```
cat > /etc/sysconfig/console << "EOF"
# Início do /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# Fim do /etc/sysconfig/console
EOF
```

- O seguinte é um exemplo habilitado para Unicode para búlgaro, onde um mapa padrão de teclas UTF-8 existe:

```
cat > /etc/sysconfig/console << "EOF"
# Início do /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# Fim do /etc/sysconfig/console
EOF
```

- Devido ao uso de uma fonte LatArCyrHeb-16 de 512 glifos no exemplo anterior, cores brilhantes não mais estão disponíveis no console do Linux, a menos que uma parte da RAM usada para armazenamento temporário de dados que estão esperando para serem enviados para um dispositivo e que armazene o conteúdo de uma imagem pixel por pixel seja usada. Se alguém quiser ter cores brilhantes sem uma parte da RAM usada para armazenamento temporário de dados que estão esperando para serem enviados para um dispositivo e que armazene o conteúdo de uma imagem pixel por pixel e puder viver sem caracteres que não pertencem a seu idioma, [então] ainda é possível usar uma fonte de 256 glifos específica para o idioma, conforme ilustrado abaixo:

```
cat > /etc/sysconfig/console << "EOF"
# Início do /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# Fim do /etc/sysconfig/console
EOF
```

- O seguinte exemplo ilustra conversão automática de mapa de teclas de ISO-8859-15 para UTF-8 e habilitação de teclas mortas em modo Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Início do /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# Fim do /etc/sysconfig/console
EOF
```

- Alguns mapas de teclas tem teclas mortas (isto é, teclas que não produzem um carácter por elas próprias, mas põem um acento no carácter produzido pela próxima tecla) ou definem regras de composição (tais como: “pressione Ctrl+. A E para obter Æ” no mapa de teclas padrão). O Linux-6.10.5 interpreta teclas mortas e regras de composição no mapa de teclas corretamente somente quando os caracteres fonte a serem compostos

juntos não são multi byte. Essa deficiência não afeta mapas de teclas para idiomas europeus, pois lá acentos são adicionados a caracteres "ASCII" não acentuados, ou dois caracteres "ASCII" são compostos juntos. Entretanto, em modo "UTF-8" isso é um problema; por exemplo, para o idioma grego, onde alguém de vez em quando precisa colocar um acento na letra α . A solução é a de, ou evitar o uso de "UTF-8", ou instalar o sistema de janelas "X", que não tem essa limitação no manuseio de entradas geradas dele.

- Para chinês, japonês, coreano e alguns outros idiomas, o console do Linux não pode ser configurado para exibir os caracteres necessários. Usuários(as) que precisam de tais idiomas deveriam instalar o Sistema de Janelas X, fontes que cobrem os intervalos necessários de caracteres, e o método de entrada adequado (por exemplo, SCIM, suporta uma ampla variedade de idiomas).



Nota

O arquivo `/etc/sysconfig/console` somente controla a localização do console de texto do Linux. Ele não tem nada a ver com configurar o esquema adequado de teclado e fontes de terminal no Sistema de Janelas X; com sessões do ssh; ou com um console serial. Em tais situações, as limitações mencionadas nos últimos dois itens de lista acima não se aplicam.

9.6.6. Criando Arquivos na Inicialização

De vez em quando, é desejável criar arquivos em tempo de inicialização. Por exemplo, o diretório `/tmp/.ICE-unix` frequentemente é necessário. Isso pode ser feito criando-se uma entrada no script de configuração `/etc/sysconfig/createfiles`. O formato desse arquivo está embutido nos comentários do arquivo padrão de configuração.

9.6.7. Configurando o Script Syslogd

O script `syslogd` invoca o aplicativo **syslogd** como uma parte da inicialização do System V. A opção `-m 0` desliga a marca de carimbo de tempo periódica que o **syslogd** escreve nos arquivos de registro a cada 20 minutos por padrão. Se você quiser ligar essa marca de carimbo de tempo periódica, [então] edite `/etc/sysconfig/rc.site` e defina a variável `SYSKLOGD_PARMS` para o valor desejado. Por exemplo, para remover todos os parâmetros, configure a variável para um valor nulo:

```
SYSKLOGD_PARMS=
```

Veja-se `man syslogd` para mais opções.

9.6.8. O Arquivo rc.site

O arquivo opcional `/etc/sysconfig/rc.site` contém configurações que são automaticamente configuradas para cada script de inicialização do SystemV. Ele pode alternativamente configurar os valores especificados nos arquivos `hostname`, `console` e `clock` no diretório `/etc/sysconfig/`. Se as variáveis associadas estiverem presentes em ambos desses arquivos separados e `rc.site`, [então] os valores nos arquivos específicos de script tem precedência.

`rc.site` também contém parâmetros que podem personalizar outros aspectos do processo de inicialização. Configurar a variável `IPROMPT` habilitará a execução seletiva de scripts de inicialização. Outras opções estão descritas nos comentários de arquivo. A versão padrão do arquivo é como se segue:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen
```

```

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\033[1;34m" # Blue
#FAILURE="\033[1;31m" # Red
#INFO="\033[1;36m" # Cyan
#NORMAL="\033[0;39m" # Grey
#SUCCESS="\033[1;32m" # Green
#WARNING="\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX=" "
#SUCCESS_PREFIX="${SUCCESS} * ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually set the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3" # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

```

```
# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m 8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. Personalizando os Scripts de Inicialização e Desligamento

Os scripts de inicialização do LFS inicializam e desligam um sistema de uma maneira bastante eficiente, porém existem uns poucos ajustes que você pode fazer no arquivo `rc.site` para aumentar a velocidade ainda mais e ajustar mensagens de acordo com suas preferências. Para fazer isso, ajuste as configurações no arquivo `/etc/sysconfig/rc.site` acima.

- Durante o script de inicialização `udev`, existe uma chamada para **udev settle** que exige algum tempo para completar. Esse tempo possivelmente ou possivelmente não seja exigido dependendo dos dispositivos no sistema. Se você tiver somente partições simples e uma placa ethernet, [então] o processo de inicialização provavelmente não precisará esperar por esse comando. Para pulá-lo, configure a variável `OMIT_UDEV_SETTLE=y`.
- O script de inicialização `udev_retry` também executa **udev settle** por padrão. Esse comando é necessário somente se o diretório `/var` for montado separadamente, pois o relógio precisa do arquivo `/var/lib/hwclock/adjtime`. Outras personalizações possivelmente também precisem esperar que o Udev complete, porém em muitas instalações ele não é necessário. Pule o comando configurando a variável `OMIT_UDEV_RETRY_SETTLE=y`.
- Por padrão, as verificações do sistema de arquivos são silenciosas. Isso pode parecer um atraso durante o processo de inicialização. Para ligar a saída gerada do **fsck**, configure a variável `VERBOSE_FSCK=y`.
- Quando reinicializar, você possivelmente queira pular a verificação do sistema de arquivos, **fsck**, completamente. Para fazer isso, ou crie o arquivo `/fastboot` ou reinicialize o sistema com o comando `/sbin/shutdown -f -r now`. Por outro lado, você pode forçar que todos os sistemas de arquivos sejam verificados criando `/forcefsck` ou executando **shutdown** com o parâmetro `-F` em vez de `-f`.

Configurar a variável `FASTBOOT=y` desabilitará **fsck** durante o processo de inicialização até que ela seja removida. Isso não é recomendado em uma base permanente.

- Normalmente, todos os arquivos no diretório `/tmp` são deletados em tempo de inicialização. Dependendo do número de arquivos ou diretórios presentes, isso pode causar um atraso notável no processo de inicialização. Para pular a remoção desses arquivos configure a variável `SKIPTMPCLEAN=y`.
- Durante o desligamento, o aplicativo **init** envia um sinal `TERM` para cada aplicativo que ele tenha iniciado (por exemplo `agetty`), espera um tempo configurado (padrão 3 segundos), então envia a cada processo um sinal `KILL` e aguarda novamente. Esse processo é repetido no script **sendsignals** para quaisquer processos que não sejam desligados pelos scripts próprios deles. O atraso para **init** pode ser configurado passando-se um parâmetro. Por exemplo, para remover o atraso no **init**, passe o parâmetro `-t0` quando desligar ou reinicializar (por exemplo `/sbin/shutdown -t0 -r now`). O atraso para o script **sendsignals** pode ser pulado configurando-se o parâmetro `KILLDELAY=0`.

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.

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 bem-sucedido, 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 a entrada do(a) usuário(a) entre os comandos.

Crie o `/etc/profile` depois que as configurações adequadas de localidade tiverem sido determinadas para configurar a localidade desejada, mas defina a localidade `C.UTF-8` se executar no console do Linux (para evitar que aplicativos 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
  export LANG=<ll>_<CC>.<mapa_caracteres><@modificadores>
fi

# Termina /etc/profile
EOF
```

As localidades `C` (padrão) e `en_US` (aquele recomendado para usuários(as) 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) usuário(a) 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 muitos outros aplicativos.

A maioria das pessoas não necessita de funcionalidade específica de usuário(a), de forma 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 usuário(a), [então] você pode criar um arquivo `.inputrc` no diretório lar do(a) usuário(a) 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"
# Início do /etc/inputrc
# Modificado por Chris Lynn <roryo@roryo.dynup.net>

# Permite ao prompt de comando passar para a próxima linha
set horizontal-scroll-mode Off

# Habilita entrada gerada de 8 bits
set meta-flag On
set input-meta On

# Desliga o despojamento do oitavo bit
set convert-meta Off

# Mantém o oitavo bit para exibir
set output-meta On

# nada, visível ou audível
set bell-style none

# Tudo do seguinte mapeia a sequência de escape do valor contido no
# primeiro argumento para as funções específicas do readline
"\eOd": backward-word
"\eOc": forward-word

# Para o console do Linux
"\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

# Para o xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# Para o Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# Fim do /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(a) desprivilegiado(a) possa mudar o shell de login para a própria conta dele(a). Se o nome de comando não estiver listado, [então] o(a) usuário(a) 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 proibem acesso a usuários(as) com shells não incluídos nesse arquivo.

```
cat > /etc/shells << "EOF"
# Início do /etc/shells

/bin/sh
/bin/bash

# Fim do /etc/shells
EOF
```

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 /etc/fstab

# sistema de arquivos ponto de montagem tipo opções despejo ordem de fsck
#

/dev/<xxx> / <fff> defaults 1 1
/dev/<yyy> swap swap pri=1 0 0
proc /proc proc nosuid,noexec,nodev 0 0
sysfs /sys sysfs nosuid,noexec,nodev 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
tmpfs /run tmpfs defaults 0 0
devtmpfs /dev devtmpfs mode=0755,nosuid 0 0
tmpfs /dev/shm tmpfs nosuid,nodev 0 0
cgroup2 /sys/fs/cgroup cgroup2 nosuid,noexec,nodev 0 0

# Fim /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 locais não UTF-8, o valor de `iocharset` deveria ser configurado para ser o mesmo que o conjunto de caracteres do locale, 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) tenha sido compilada no núcleo ou construída como um módulo. Entretanto, se o conjunto de caracteres do locale for UTF-8, [então] 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 locais 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 seu país. Por exemplo, para a finalidade de montar unidades USB flash, um(a) usuário(a) do ru_RU.KOI8-R precisaria do seguinte na porção de opções da linha `mount` dele em `/etc/fstab`:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

O correspondente fragmento das opções para usuários(as) 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.

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 chamados 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.

É possível tornar o sistema de arquivos ext3 confiável em casos de falhas de eletricidade para alguns tipos de disco rígido. Para fazer isso, adicione a opção de montagem `barrier=1` à entrada apropriada em `/etc/fstab`. Para verificar se a unidade de disco suporta essa opção, execute `hdparm` na unidade de disco aplicável. Por exemplo, se:

```
hdparm -I /dev/sda | grep NCQ
```

retornar saída gerada não vazia, [então] a opção é suportada.

Observe: partições baseadas em Logical Volume Management (LVM) não podem usar a opção `barrier`.

10.3. Linux-6.10.5

O pacote Linux contém o núcleo Linux.

Tempo aproximado de 0,4 - 32 UPC (tipicamente cerca de 2,5 UPC)

construção:

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) usuários(as) 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://andu.in.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 do locale 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, [então] assegure-se de substituir `<host_LANG_value>` pelo valor da variável `$LANG` oriunda do seu anfitrião. Você pode, alternativamente, usar, em vez disso, o valor do anfitrião de `$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]
[ ] 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]

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]

Graphics support --->
< /*/M> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
                                     ... [DRM]

# If [DRM] is selected as * or M, this must be selected:
[ /*] Enable legacy fbdev support for your modesetting driver
                                     ... [DRM_FBDEV_EMULATION]

Console display driver support --->
# If [DRM] is selected as * or M, this must be selected:
[ /*] Framebuffer Console support                            [FRAMEBUFFER_CONSOLE]

```

Habilite alguns recursos adicionais se você estiver construindo um sistema de 64 bits. Se você estiver usando o menuconfig, [então] 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 --->
[*] Support x2apic                                          [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 executando em um hardware com RAM mais que 4 GB, ajuste a configuração, de modo que o núcleo consiga usar até 64 GB de RAM física:

```

Processor type and features --->
High Memory Support --->
(X) 64GB                                                    [HIGHMEM64G]

```

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]
```

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*.



Nota

Se seu hardware do anfitrião estiver usando UEFI e você desejar inicializar o sistema LFS com ela, [então] você deveria ajustar alguma configuração do núcleo seguindo *a página do BLFS*, **mesmo se você usará o carregador de inicialização UEFI da distribuição anfitriã**.

A justificativa para os itens de configuração acima:

Torne aleatório o endereço da imagem do núcleo (KASLR)

Habilita ASLR para imagem do núcleo, para mitigar alguns ataques baseados em endereços fixos de dados ou código sensíveis no núcleo.

Compila o núcleo com avisos como erros

Isso possivelmente cause falha de construção se o compilador e (ou) a configuração forem diferentes daqueles dos(as) desenvolvedores(as) do núcleo.

Habilita cabeçalhos do núcleo por meio de /sys/kernel/kheaders.tar.xz

Isso exigirá **cpio** ao se construir o núcleo. **cpio** não é instalado pelo LFS.

Configurar recursos padrão do núcleo (usuários(as) experientes)

Isso fará com que algumas opções apareçam na interface de configuração, mas alterar essas opções possivelmente seja perigoso. Não use isso, a menos que você saiba o que está fazendo.

Protetor Forte da Pilha

Habilita SSP para o núcleo. Nós o habilitamos para o espaço inteiro de usuário(a) 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.

Suporte a auxiliar do uevent

Ter essa opção configurada possivelmente interfira com o gerenciamento de dispositivo quando se usar o Udev.

Mantém um 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(as) do Udev.

Automonta devtmpfs em /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.

Enable legacy fbdev support for your modesetting driver e Framebuffer Console support

Esses são necessários para exibir o console do Linux em uma "GPU" controlada por um controlador "Direct Rendering Infrastructure" ("DRI"). Se "CONFIG_DRM" ("Direct Rendering Manager") estiver habilitado, [então] você deveria habilitar essas duas opções também ou verá uma tela em branco quando o controlador "DRI" for carregado.

Suporte a 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, [então] pule a configuração do núcleo copiando o arquivo `config` do núcleo, `.config`, a partir do sistema anfitrião (assumindo que ele esteja disponível) para o diretório `linux-6.10.5` 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, [então] 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-6.10.5/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), [então] 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) usuário(a) `root` dentro do *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 `vmlinuz` 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/vmlinuz-6.10.5-lfs-12.2
```

`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-6.10.5
```

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-6.10.5
```

Instale a documentação para o núcleo Linux:

```
cp -r Documentation -T /usr/share/doc/linux-6.10.5
```

É importante observar que os arquivos no diretório do fonte do núcleo não são de propriedade do(a) *root*. Sempre que um pacote é descompactado como o(a) usuário(a) *root* (como nós fizemos dentro do *chroot*), os arquivos tem os IDs de usuário(a) 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 é 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 usuário(a) 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, [então] execute **chown -R 0:0** no diretório `linux-6.10.5` para assegurar que todos os arquivos sejam de propriedade do(a) usuário(a) *root*.



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.



Atenção

Os cabeçalhos no diretório `include` do sistema (`/usr/include`) deveriam *sempre* ser aqueles contra os quais a Glibc foi compilada, isto é, os cabeçalhos sanitizados instalados na Seção 5.4, “Cabeçalhos da API do Linux-6.10.5”. Portanto, eles *nunca* deveriam ser substituídos nem pelos cabeçalhos crus do núcleo nem por quaisquer outros cabeçalhos sanitizados do núcleo.

10.3.2. Configurando a Ordem de Carregamento de Módulo do Linux

Na maior parte do tempo, os módulos do Linux são carregados automaticamente, porém algumas vezes precisa-se de alguma direção específica. O aplicativo que carrega os módulos, **modprobe** ou o **insmod**, usa `/etc/modprobe.d/usb.conf` para esse propósito. Esse arquivo precisa ser criado, de forma que, se os controladores do USB (`ehci_hcd`, `ohci_hcd` e `uhci_hcd`) tiverem sido construídos como módulos, [então] eles sejam carregados na ordem correta; `ehci_hcd` precisa ser carregado antes de `ohci_hcd` e `uhci_hcd` para a finalidade de evitar um aviso sendo produzido em tempo de inicialização.

Crie um novo arquivo `/etc/modprobe.d/usb.conf` executando o seguinte:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Início do /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# Fim do /etc/modprobe.d/usb.conf
EOF
```

10.3.3. Conteúdo do Linux

Arquivos instalados: `config-6.10.5`, `vmlinuz-6.10.5-lfs-12.2` e `System.map-6.10.5`
Diretórios instalados: `/lib/modules` e `/usr/share/doc/linux-6.10.5`

Descrições Curtas

`config-6.10.5`

Contém todas as seleções de configuração para o núcleo

`vmlinuz-6.10.5-lfs-12.2`

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-6.10.5`

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



Nota

Se o seu sistema tem suporte UEFI e você deseja inicializar o LFS com UEFI, [então] você deveria pular as instruções nesta página, mas ainda assim aprender a sintaxe do `grub.cfg` e o método para especificar uma partição no arquivo a partir desta página e configurar o GRUB com suporte UEFI usando as instruções fornecidas na *página BLFS*.

10.4.1. Introdução



Atenção

Configurar o GRUB incorretamente pode tornar seu sistema inoperável sem um dispositivo alternativo de inicialização, como um CD-ROM ou unidade USB inicializável. Esta seção não é exigida para inicializar seu sistema LFS. Você possivelmente apenas queira modificar seu carregador de inicialização atual, por exemplo, Grub-Legacy, GRUB2 ou LILO.

Certifique-se de que um disco de inicialização de emergência esteja pronto para “resgatar” o computador se o computador se tornar inutilizável (não inicializável). Se você ainda não tiver um dispositivo de inicialização, [então] você pode criar um. Para a finalidade de que o procedimento abaixo funcione, você precisa saltar para a frente 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. Convenções de Nomenclatura do GRUB

O GRUB usa estrutura de nomenclatura própria dele para unidades e partições na forma de (hdn,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`, [então] aquela segunda unidade rígida ainda seria $(hd1)$.

10.4.3. Definindo a Configuração

O GRUB funciona escrevendo dados na primeira trilha física do disco rígido. Essa área não é parte de nenhum sistema de arquivos. Os aplicativos lá acessam módulos do GRUB na partição de inicialização. O local padrão é `/boot/grub/`.

O local da partição de inicialização é uma escolha do(a) usuário(a) que afeta a configuração. Uma recomendação é ter uma partição pequena (tamanho sugerido é 200 MB) separada somente para informação de inicialização. Dessa forma, cada construção, seja LFS ou alguma distribuição comercial, consegue acessar os mesmos arquivos de inicialização e o acesso pode ser feito a partir de qualquer sistema inicializado. Se você escolher fazer isso, [então] você precisará montar a partição separada, mover todos os arquivos no diretório `/boot` atual (por exemplo, o núcleo Linux que você recém construiu na seção anterior) para a nova partição. Você então precisará desmontar a partição e remontá-la como `/boot`. Se você fizer isso, [então] tenha certeza de atualizar `/etc/fstab`.

Deixar `/boot` na partição LFS atual também funcionará, porém a configuração para múltiplos sistemas é mais complicada.

Usando a informação 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`.

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 para gerenciar o Master Boot Record (MBR).

```
grub-install /dev/sda
```



Nota

Se o sistema tiver sido inicializado usando UEFI, [então] o **grub-install** tentará instalar arquivos para o alvo `x86_64-efi`, porém aqueles arquivos não foram instalados no Capítulo 8. Se esse for o caso, [então] adicione `--target i386-pc` ao comando acima.

10.4.4. 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)

menuentry "GNU/Linux, Linux 6.10.5-lfs-12.2" {
    linux /boot/vmlinuz-6.10.5-lfs-12.2 root=/dev/sda2 ro
}
EOF
```

Os comandos **insmod** carregam os módulos GRUB chamados `part_gpt` e `ext2`. Apesar da nomenclatura, `ext2` na verdade suporta sistemas de arquivos `ext2`, `ext3` e `ext4`. O comando **grub-install** incorporou alguns módulos na imagem principal do GRUB (instalada no "MBR" ou na partição "BIOS" do GRUB) para acessar os outros módulos (em `/boot/grub/i386-pc`) sem um problema de galinha ou ovo, de modo que, com uma configuração típica, esses dois módulos já estão incorporados e esses dois comandos **insmod** não farão nada. Mas eles não fazem mal de qualquer maneira e possivelmente sejam necessários com algumas configurações raras.



Nota

A partir da perspectiva do GRUB, os arquivos do núcleo estão relativos à partição usada. Se você usou uma partição `/boot` separada, [então] 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 você deseja evitar tal problema, [então] 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 seus sistemas de arquivos (na coluna `UUID`) e das suas partições (na coluna `PARTUUID`). Então substitua `set root=(hdx,y)` por `search --set=root --fs-uuid <UUID do sistema de arquivos onde o núcleo está instalado>`; e substitua `root=/dev/sda2` por `root=PARTUUID=<UUID da partição onde o LFS está 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 (isso é menos provável que uma mudança do designador do GRUB). Você também pode substituir caminhos para nós de dispositivo, como `/dev/sda1`, por `PARTUUID=<UUID da partição>`, em `/etc/fstab`, para evitar uma potencial falha de inicialização no caso do nome do nó de dispositivo tiver mudado.

O GRUB é um aplicativo 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, [então] existe uma boa chance de que esse aplicativo seja executado. Tenha certeza de produzir uma cópia de segurança do seu 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 12.2 > /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="12.2"
DISTRIB_CODENAME="<seu nome aqui>"
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="12.2"
ID=lfs
PRETTY_NAME="Linux From Scratch 12.2"
VERSION_CODENAME="<seu nome aqui>"
HOME_URL="https://www.linuxfromscratch.org/lfs/"
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 você terminou o livro, você quer ser contado(a) como um(a) usuário(a) do LFS? Vá para <https://www.linuxfromscratch.org/cgi-bin/lfscounter.php> e registre-se como um(a) usuário(a) do LFS fornecendo seu nome e a primeira versão do LFS que você usou.

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 definida para o(a) usuário(a) `root`.
- Uma revisão dos seguintes arquivos de configuração também é apropriada neste ponto.

- /etc/bashrc
- /etc/dircolors
- /etc/fstab
- /etc/hosts
- /etc/inputrc
- /etc/profile
- /etc/resolv.conf
- /etc/vimrc
- /root/.bash_profile
- /root/.bashrc
- /etc/sysconfig/ifconfig.eth0

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 12.2* 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 exige um ambiente gráfico de usuário(a), 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 todos os ambientes. 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 usuário(a) 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 script pode ser executado com **bash ~/mount-virt.sh** como ou um(a) usuário(a) comum (recomendado) ou como **root**. Se executado como um(a) usuário(a) comum, o **sudo** é exigido no sistema anfitrião.

Outro problema apontado pelo script é o de onde armazenar os arquivos baixados do pacote. Esse local é arbitrário. Ele pode estar em um diretório lar de usuário(a) 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 usuário(a) no sistema anfitrião:

```
alias lfs='sudo /usr/sbin/chroot /mnt/lfs /usr/bin/env -i HOME=/root TERM="$TERM" PS1="\u:\w\\\$ "
PATH=/bin:/usr/bin:/sbin:/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 usuário(a) 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 de tradução do LFS para francês 2008-2016
- *Julien Lepiller* <jlepiller@linuxfromscratch.org> – Projeto de tradução do LFS para francês 2017-presente
- *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 de tradução do LFS para alemão

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
- *Bastiaan 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@linuxpourtous.com> – espelho lfs.linuxpourtous.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 Phermawang* <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
- *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) usuário(a). 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 é endereçada no BLFS.

Acl

A Instalação depende de:	Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução:	Attr e 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

Check

A Instalação depende de:	Gawk, GCC, Grep, Make, Sed e Texinfo
Exigido em tempo de execução:	Bash e Gawk
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>libsubunit</i> e <i>patchutils</i>

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 Udev
Dependências opcionais:	<i>Expect.pm</i> e <i>IO::Tty</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, 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 e Zlib
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:	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 e 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, 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:	<i>PCRE2 e libsigsegv</i>

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:	Nenhuma suíte de teste disponível
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

Intltool

A Instalação depende de:	Bash, Gawk, Glibc, Make, Perl, Sed e XML::Parser
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, libbpf, 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:	Udev
Dependências opcionais:	Nenhum

Kbd

A Instalação depende de:	Bash, Binutils, Bison, Check, 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 e Zlib
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:	Udev
Dependências opcionais:	<i>scdoc</i> (para páginas de manual)

Less

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed
Exigido em tempo de execução:	Glibc e Ncurses
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Gzip
Dependências opcionais:	<i>PCRE2</i> ou <i>PCRE</i>

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:	Check e 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 Udev
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, LLVM (com Clang) e 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
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 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:	Udev
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 Udev
Dependências opcionais:	Nenhum

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>

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:	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:	Binutils, E2fsprogs, IProute2, Kmod, Man-DB, Procps-ng, Python, Udev e Util-linux
Dependências opcionais:	Nenhum

Procps-ng

A Instalação depende de:	Bash, Binutils, Coreutils, GCC, Glibc, Make, Ncurses e Pkgconf
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	DejaGNU
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	<i>elogind</i>

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:	Nenhuma suíte de teste disponível
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, Ncurses, OpenSSL, Pkgconf, Sed e Util-linux
Exigido em tempo de execução:	Bzip2, Expat, Gdbm, Glibc, Libffi, Libxcrypt, 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, SQLite 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>

Sysklogd

A Instalação depende de:	Binutils, Coreutils, GCC, Glibc, Make e Patch
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

SysVinit

A Instalação depende de:	Binutils, Coreutils, GCC, Glibc, Make e Sed
Exigido em tempo de execução:	Glibc
A suíte de teste depende de:	Nenhuma suíte de teste disponível
Precisa ser instalado antes de:	Nenhum
Dependências opcionais:	Nenhum

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:	Nenhum

Udev

A Instalação depende de:	Acl, Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libcap, Libxcrypt, Meson, OpenSSL, Pkgconf, Sed, Util-linux e Zstd
Exigido em tempo de execução:	Acl, Glibc, Libcap, OpenSSL e Util-linux
A suíte de teste depende de:	Nenhum
Precisa ser instalado antes de:	Util-linux
Dependências opcionais:	Nenhum

Util-linux

A Instalação depende de:	Bash, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Pkgconf, Sed, Udev e Zlib
Exigido em tempo de execução:	Glibc, Ncurses, Readline, Udev 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 e Flit-core
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

XML::Parser

A Instalação depende de:	Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make e Perl
Exigido em tempo de execução:	Expat, Glibc e Perl
A suíte de teste depende de:	Perl
Precisa ser instalado antes de:	Intltool
Dependências opcionais:	Nenhum

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 Udev
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 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, GCC, Libelf e Udev
Dependências opcionais:	Nenhum

Apêndice D. Scripts de inicialização e configuração do sistema versão-20240825

Os scripts neste anexo estão listados pelo diretório onde eles normalmente residem. A ordem é `/etc/rc.d/init.d`; `/etc/sysconfig`; `/etc/sysconfig/network-devices`; e `/etc/sysconfig/network-devices/services`. Dentro de cada seção, os arquivos estão listados na ordem em que eles normalmente são chamados.

D.1. `/etc/rc.d/init.d/rc`

O script `rc` é o primeiro script chamado pelo `init` e inicia o processo de inicialização.

```
#!/bin/bash
#####
# Begin rc
#
# Description : Main Run Level Control Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               : DJ Lucas - dj@linuxfromscratch.org
# Updates      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : Updates March 24th, 2022: new semantics of S/K files
#               - Instead of testing that S scripts were K scripts in the
#                 previous runlevel, test that they were not S scripts
#               - Instead of testing that K scripts were S scripts in the
#                 previous runlevel, test that they were not K scripts
#               - S scripts in runlevel 0 or 6 are now run with
#                 "script start" (was "script stop" previously).
#####

. /lib/lsb/init-functions

print_error_msg()
{
    log_failure_msg
    # $i is set when called
    MSG="FAILURE:\n\nYou should not be reading this error message.\n\n"
    MSG="${MSG}It means that an unforeseen error took place in\n"
    MSG="${MSG}${i},\n"
    MSG="${MSG}which exited with a return value of ${error_value}.\n"

    MSG="${MSG}If you're able to track this error down to a bug in one of\n"
    MSG="${MSG}the files provided by the ${DISTRO_MINI} book,\n"
    MSG="${MSG}please be so kind to inform us at ${DISTRO_CONTACT}.\n"
    log_failure_msg "${MSG}"

    log_info_msg "Press Enter to continue..."
    wait_for_user
}

check_script_status()
{
    # $i is set when called
    if [ ! -f ${i} ]; then
        log_warning_msg "${i} is not a valid symlink."
        SCRIPT_STAT="1"
    fi

    if [ ! -x ${i} ]; then
        log_warning_msg "${i} is not executable, skipping."
    fi
}
```

```

    SCRIPT_STAT="1"
    fi
}
run()
{
    if [ -z $interactive ]; then
        ${1} ${2}
        return $?
    fi

    while true; do
        read -p "Run ${1} ${2} (Yes/no/continue)? " -n 1 runit
        echo

        case ${runit} in
            c | C)
                interactive=""
                ${i} ${2}
                ret=${?}
                break;
                ;;

            n | N)
                return 0
                ;;

            y | Y)
                ${i} ${2}
                ret=${?}
                break
                ;;

            esac
        done

        return $ret
    }

# Read any local settings/overrides
[ -r /etc/sysconfig/rc.site ] && source /etc/sysconfig/rc.site

DISTRO=${DISTRO:-"Linux From Scratch"}
DISTRO_CONTACT=${DISTRO_CONTACT:-"lfs-dev@lists.linuxfromscratch.org (Registration required)"}
DISTRO_MINI=${DISTRO_MINI:-"LFS"}
IPROMPT=${IPROMPT:-"no"}

# These 3 signals will not cause our script to exit
trap "" INT QUIT TSTP

[ "${1}" != "" ] && runlevel=${1}

if [ "${runlevel}" == "" ]; then
    echo "Usage: ${0} <runlevel>" >&2
    exit 1
fi

previous=${PREVLEVEL}
[ "${previous}" == "" ] && previous=N

if [ ! -d /etc/rc.d/rc${runlevel}.d ]; then
    log_info_msg "/etc/rc.d/rc${runlevel}.d does not exist.\n"
    exit 1
fi

if [ "$runlevel" == "6" -o "$runlevel" == "0" ]; then IPROMPT="no"; fi

# Note: In ${LOGLEVEL:-7}, it is ':' 'dash' '7', not minus 7

```

```

if [ "$runlevel" == "S" ]; then
    [ -r /etc/sysconfig/console ] && source /etc/sysconfig/console
    dmesg -n "${LOGLEVEL:-7}"
fi

if [ "${IPROMPT}" == "yes" -a "${runlevel}" == "S" ]; then
    # The total length of the distro welcome string, without escape codes
    wlen=${wlen:-$(echo "Welcome to ${DISTRO}" | wc -c )}
    welcome_message=${welcome_message:-"Welcome to ${INFO}${DISTRO}${NORMAL}"}

    # The total length of the interactive string, without escape codes
    ilen=${ilen:-$(echo "Press 'I' to enter interactive startup" | wc -c )}
    i_message=${i_message:-"Press '${FAILURE}I${NORMAL}' to enter interactive startup"}

    # dcol and icol are spaces before the message to center the message
    # on screen. itime is the amount of wait time for the user to press a key
    wcol=$(( ( ${COLUMNS} - ${wlen} ) / 2 ))
    icol=$(( ( ${COLUMNS} - ${ilen} ) / 2 ))
    itime=${itime:-"3"}

    echo -e "\n\n"
    echo -e "\\033[${wcol}G${welcome_message}"
    echo -e "\\033[${icol}G${i_message}${NORMAL}"
    echo ""
    read -t "${itime}" -n 1 interactive 2>&1 > /dev/null
fi

# Make lower case
[ "${interactive}" == "I" ] && interactive="i"
[ "${interactive}" != "i" ] && interactive=""

# Read the state file if it exists from runlevel S
[ -r /run/interactive ] && source /run/interactive

# Stop all services marked as K, except if marked as K in the previous
# runlevel: it is the responsibility of the script to not try to kill
# a non running service
if [ "${previous}" != "N" ]; then
    for i in $(ls -v /etc/rc.d/rc${runlevel}.d/K* 2> /dev/null)
    do
        check_script_status
        if [ "${SCRIPT_STAT}" == "1" ]; then
            SCRIPT_STAT="0"
            continue
        fi

        suffix=${i#/etc/rc.d/rc${runlevel}.d/K[0-9][0-9]}
        [ -e /etc/rc.d/rc${previous}.d/K[0-9][0-9]$suffix ] && continue

        run ${i} stop
        error_value=${?}

        if [ "${error_value}" != "0" ]; then print_error_msg; fi
    done
fi

if [ "${previous}" == "N" ]; then export IN_BOOT=1; fi

if [ "$runlevel" == "6" -a -n "${FASTBOOT}" ]; then
    touch /fastboot
fi

# Start all services marked as S in this runlevel, except if marked as
# S in the previous runlevel
# it is the responsibility of the script to not try to start an already running

```

```

# service
for i in $( ls -v /etc/rc.d/rc${runlevel}.d/S* 2> /dev/null )
do

    if [ "${previous}" != "N" ]; then
        suffix=${i#/etc/rc.d/rc${runlevel}.d/S[0-9][0-9]}
        [ -e /etc/rc.d/rc${previous}.d/S[0-9][0-9]$suffix ] && continue
    fi

    check_script_status
    if [ "${SCRIPT_STAT}" == "1" ]; then
        SCRIPT_STAT="0"
        continue
    fi

    run ${i} start

    error_value=${?}

    if [ "${error_value}" != "0" ]; then print_error_msg; fi
done

# Store interactive variable on switch from runlevel S and remove if not
if [ "${runlevel}" == "S" -a "${interactive}" == "i" ]; then
    echo "interactive=\"i\"" > /run/interactive
else
    rm -f /run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a "${runlevel}" != "S" ]; then
    cat $BOOTLOG >> /var/log/boot.log

    # Mark the end of boot
    echo "-----" >> /var/log/boot.log

    # Remove the temporary file
    rm -f $BOOTLOG 2> /dev/null
fi

# End rc

```

D.2. /lib/lsb/init-functions

```

#!/bin/sh
#####
#
# Begin /lib/lsb/init-funtions
#
# Description : Run Level Control Functions
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
# Notes       : With code based on Matthias Benkmann's simpleinit-msb
#              http://winterdrache.de/linux/newboot/index.html
#
#              The file should be located in /lib/lsb
#
#####

## Environmental setup
# Setup default values for environment
umask 022

```

```

export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

## Set color commands, used via echo
# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (l;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles

NORMAL="\033[0;39m"      # Standard console grey
SUCCESS="\033[1;32m"     # Success is green
WARNING="\033[1;33m"     # Warnings are yellow
FAILURE="\033[1;31m"     # Failures are red
INFO="\033[1;36m"       # Information is light cyan
BRACKET="\033[1;34m"    # Brackets are blue

# Use a colored prefix
BMPREFIX=""
SUCCESS_PREFIX="${SUCCESS} * ${NORMAL} "
FAILURE_PREFIX="${FAILURE} ***** ${NORMAL} "
WARNING_PREFIX="${WARNING} *** ${NORMAL} "
SKIP_PREFIX="${INFO} S ${NORMAL} "

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS} OK ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"
SKIP_SUFFIX="${BRACKET}[${INFO} SKIP ${BRACKET}]${NORMAL}"

BOOTLOG=/run/bootlog
KILLDELAY=3
SCRIPT_STAT="0"

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ] && . /etc/sysconfig/rc.site

# If HEADLESS is set, use that.
# If file descriptor 1 or 2 (stdout and stderr) is not open or
# does not refer to a terminal, consider the script headless.
[ ! -t 1 -o ! -t 2 ] && HEADLESS=${HEADLESS:-yes}

if [ "x$HEADLESS" != "xyes" ]
then
  ## Screen Dimensions
  # Find current screen size
  if [ -z "${COLUMNS}" ]; then
    COLUMNS=$(stty size)
    COLUMNS=${COLUMNS##* }
  fi
else
  COLUMNS=80
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
  COLUMNS=80
fi

## Measurements for positioning result messages
COL=$(( ${COLUMNS} - 8 ))
WCOL=$(( ${COL} - 2 ))

## Set Cursor Position Commands, used via echo
SET_COL="\033[${COL}G"      # at the $COL char
SET_WCOL="\033[${WCOL}G"    # at the $WCOL char
CURS_UP="\033[1A\033[0G"   # Up one line, at the 0'th char

```

```

CURS_ZERO="\033[0G"

#####
# start_daemon() #
# Usage: start_daemon [-f] [-n nicelevel] [-p pidfile] pathname [args...] #
# # #
# Purpose: This runs the specified program as a daemon #
# # #
# Inputs: -f: (force) run the program even if it is already running. #
# -n nicelevel: specify a nice level. See 'man nice(1)'. #
# -p pidfile: use the specified file to determine PIDs. #
# pathname: the complete path to the specified program #
# args: additional arguments passed to the program (pathname) #
# # #
# Return values (as defined by LSB exit codes): #
# 0 - program is running or service is OK #
# 1 - generic or unspecified error #
# 2 - invalid or excessive argument(s) #
# 5 - program is not installed #
#####
start_daemon()
{
    local force=""
    local nice="0"
    local pidfile=""
    local pidlist=""
    local retval=""

    # Process arguments
    while true
    do
        case "${1}" in

            -f)
                force="1"
                shift 1
                ;;

            -n)
                nice="${2}"
                shift 2
                ;;

            -p)
                pidfile="${2}"
                shift 2
                ;;

            -*)
                return 2
                ;;

            *)
                program="${1}"
                break
                ;;
        esac
    done

    # Check for a valid program
    if [ ! -e "${program}" ]; then return 5; fi

    # Execute
    if [ -z "${force}" ]; then
        if [ -z "${pidfile}" ]; then
            # Determine the pid by discovery
            pidlist=`pidofproc "${1}"`

```

```

    retval="${?}"
else
    # The PID file contains the needed PIDs
    # Note that by LSB requirement, the path must be given to pidofproc,
    # however, it is not used by the current implementation or standard.
    pidlist=`pidofproc -p "${pidfile}" "${1}"`
    retval="${?}"
fi

# Return a value ONLY
# It is the init script's (or distribution's functions) responsibility
# to log messages!
case "${retval}" in

    0)
        # Program is already running correctly, this is a
        # successful start.
        return 0
        ;;

    1)
        # Program is not running, but an invalid pid file exists
        # remove the pid file and continue
        rm -f "${pidfile}"
        ;;

    3)
        # Program is not running and no pidfile exists
        # do nothing here, let start_daemon continue.
        ;;

    *)
        # Others as returned by status values shall not be interpreted
        # and returned as an unspecified error.
        return 1
        ;;

esac

fi

# Do the start!
nice -n "${nice}" "${@"}"
}

#####
# killproc()
# Usage: killproc [-p pidfile] pathname [signal]
#
# Purpose: Send control signals to running processes
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Return values (as defined by LSB exit codes):
#     0 - program (pathname) has stopped/is already stopped or a
#         running program has been sent specified signal and stopped
#         successfully
#     1 - generic or unspecified error
#     2 - invalid or excessive argument(s)
#     5 - program is not installed
#     7 - program is not running and a signal was supplied
#####
killproc()
{
    local pidfile
    local program
    local prefix

```

```

local progname
local signal="-TERM"
local fallback="-KILL"
local nosig
local pidlist
local retval
local pid
local delay="30"
local piddead
local dtime

# Process arguments
while true; do
    case "${1}" in
        -p)
            pidfile="${2}"
            shift 2
            ;;

        *)
            program="${1}"
            if [ -n "${2}" ]; then
                signal="${2}"
                fallback=""
            else
                nosig=1
            fi

            # Error on additional arguments
            if [ -n "${3}" ]; then
                return 2
            else
                break
            fi
            ;;
    esac
done

# Check for a valid program
if [ ! -e "${program}" ]; then return 5; fi

# Check for a valid signal
check_signal "${signal}"
if [ "${?}" -ne "0" ]; then return 2; fi

# Get a list of pids
if [ -z "${pidfile}" ]; then
    # determine the pid by discovery
    pidlist=`pidofproc "${1}"`
    retval="${?}"
else
    # The PID file contains the needed PIDs
    # Note that by LSB requirement, the path must be given to pidofproc,
    # however, it is not used by the current implementation or standard.
    pidlist=`pidofproc -p "${pidfile}" "${1}"`
    retval="${?}"
fi

# Return a value ONLY
# It is the init script's (or distribution's functions) responsibility
# to log messages!
case "${retval}" in

    0)
        # Program is running correctly
        # Do nothing here, let killproc continue.
        ;;

```

```

1)
# Program is not running, but an invalid pid file exists
# Remove the pid file.

progrname=${program##*/}

if [[ -e "/run/${progrname}.pid" ]]; then
    pidfile="/run/${progrname}.pid"
    rm -f "${pidfile}"
fi

# This is only a success if no signal was passed.
if [ -n "${nosig}" ]; then
    return 0
else
    return 7
fi
;;

3)
# Program is not running and no pidfile exists
# This is only a success if no signal was passed.
if [ -n "${nosig}" ]; then
    return 0
else
    return 7
fi
;;

*)
# Others as returned by status values shall not be interpreted
# and returned as an unspecified error.
return 1
;;

esac

# Perform different actions for exit signals and control signals
check_sig_type "${signal}"

if [ "${?}" -eq "0" ]; then # Signal is used to terminate the program

    # Account for empty pidlist (pid file still exists and no
    # signal was given)
    if [ "${pidlist}" != "" ]; then

        # Kill the list of pids
        for pid in ${pidlist}; do

            kill -0 "${pid}" 2> /dev/null

            if [ "${?}" -ne "0" ]; then
                # Process is dead, continue to next and assume all is well
                continue
            else
                kill "${signal}" "${pid}" 2> /dev/null

                # Wait up to ${delay}/10 seconds to for "${pid}" to
                # terminate in 10ths of a second

                while [ "${delay}" -ne "0" ]; do
                    kill -0 "${pid}" 2> /dev/null || piddead="1"
                    if [ "${piddead}" = "1" ]; then break; fi
                    sleep 0.1
                    delay=$(( ${delay} - 1 ))
                done
            fi
        done
    fi
fi

```

```

        # If a fallback is set, and program is still running, then
        # use the fallback
        if [ -n "${fallback}" -a "${piddead}" != "1" ]; then
            kill "${fallback}" "${pid}" 2> /dev/null
            sleep 1
            # Check again, and fail if still running
            kill -0 "${pid}" 2> /dev/null && return 1
        fi
    fi
done
fi

# Check for and remove stale PID files.
if [ -z "${pidfile}" ]; then
    # Find the basename of $program
    prefix=`echo "${program}" | sed 's/[^/]*$//'`
    proname=`echo "${program}" | sed "s@${prefix}@"`

    if [ -e "/run/${proname}.pid" ]; then
        rm -f "/run/${proname}.pid" 2> /dev/null
    fi
else
    if [ -e "${pidfile}" ]; then rm -f "${pidfile}" 2> /dev/null; fi
fi

# For signals that do not expect a program to exit, simply
# let kill do its job, and evaluate kill's return for value

else # check_sig_type - signal is not used to terminate program
    for pid in ${pidlist}; do
        kill "${signal}" "${pid}"
        if [ "${?}" -ne "0" ]; then return 1; fi
    done
fi
}

#####
# pidofproc() #
# Usage: pidofproc [-p pidfile] pathname #
# # #
# Purpose: This function returns one or more pid(s) for a particular daemon #
# # #
# Inputs: -p pidfile, use the specified pidfile instead of pidof #
# # #
# # #
# Return values (as defined by LSB status codes): #
# # #
# 0 - Success (PIDs to stdout) #
# 1 - Program is dead, PID file still exists (remaining PIDs output) #
# 3 - Program is not running (no output) #
#####
pidofproc()
{
    local pidfile
    local program
    local prefix
    local proname
    local pidlist
    local lpids
    local exitstatus="0"

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2

```

```

        ;;

        *)
        program="${1}"
        if [ -n "${2}" ]; then
            # Too many arguments
            # Since this is status, return unknown
            return 4
        else
            break
        fi
        ;;
    esac
done

# If a PID file is not specified, try and find one.
if [ -z "${pidfile}" ]; then
    # Get the program's basename
    prefix=`echo "${program}" | sed 's/[^/]*$//`

    if [ -z "${prefix}" ]; then
        procname="${program}"
    else
        procname=`echo "${program}" | sed "s@${prefix}@@"`
    fi

    # If a PID file exists with that name, assume that is it.
    if [ -e "/run/${procname}.pid" ]; then
        pidfile="/run/${procname}.pid"
    fi
fi

# If a PID file is set and exists, use it.
if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
    # Use the value in the first line of the pidfile
    pidlist=`/bin/head -n1 "${pidfile}"`
else
    # Use pidof
    pidlist=`pidof "${program}"`
fi

# Figure out if all listed PIDs are running.
for pid in ${pidlist}; do
    kill -0 ${pid} 2> /dev/null

    if [ "${?}" -eq "0" ]; then
        lpids="${lpids}${pid} "
    else
        exitstatus="1"
    fi
done

if [ -z "${lpids}" -a ! -f "${pidfile}" ]; then
    return 3
else
    echo "${lpids}"
    return "${exitstatus}"
fi
}

#####
# statusproc() #
# Usage: statusproc [-p pidfile] pathname #
# # #
# Purpose: This function prints the status of a particular daemon to stdout #
# # #
# Inputs: -p pidfile, use the specified pidfile instead of pidof #

```

```

#      pathname, path to the specified program      #
#      #      #
# Return values:      #
#      0 - Status printed      #
#      1 - Input error. The daemon to check was not specified.      #
#####
statusproc()
{
    local pidfile
    local pidlist

    if [ "${#}" = "0" ]; then
        echo "Usage: statusproc [-p pidfile] {program}"
        exit 1
    fi

    # Process arguments
    while true; do
        case "${1}" in

            -p)
                pidfile="${2}"
                shift 2
                ;;

            *)
                if [ -n "${2}" ]; then
                    echo "Too many arguments"
                    return 1
                else
                    break
                fi
                ;;
        esac
    done

    if [ -n "${pidfile}" ]; then
        pidlist=`pidofproc -p "${pidfile}" $@`
    else
        pidlist=`pidofproc $@`
    fi

    # Trim trailing blanks
    pidlist=`echo "${pidlist}" | sed -r 's/ +$//`

    base="${1##*/}"

    if [ -n "${pidlist}" ]; then
        /bin/echo -e "${INFO}${base} is running with Process" \
            "ID(s) ${pidlist}.${NORMAL}"
    else
        if [ -n "${base}" -a -e "/run/${base}.pid" ]; then
            /bin/echo -e "${WARNING}${1} is not running but" \
                "/run/${base}.pid exists.${NORMAL}"
        else
            if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
                /bin/echo -e "${WARNING}${1} is not running" \
                    "but ${pidfile} exists.${NORMAL}"
            else
                /bin/echo -e "${INFO}${1} is not running.${NORMAL}"
            fi
        fi
    fi
}

#####
# timespec()      #

```

```

#                                                                                                     #
# Purpose: An internal utility function to format a timestamp                                       #
#           a boot log file.  Sets the STAMP variable.                                             #
#                                                                                                     #
# Return value: Not used                                                                              #
#####
timespec()
{
    STAMP="$(echo `date +%b %d %T %:z` `hostname`) "
    return 0
}

#####
# log_success_msg()                                                                                   #
# Usage: log_success_msg ["message"]                                                                #
#                                                                                                     #
# Purpose: Print a successful status message to the screen and                                     #
#           a boot log file.                                                                        #
#                                                                                                     #
# Inputs:  $@ - Message                                                                              #
#                                                                                                     #
# Return values: Not used                                                                            #
#####
log_success_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
        logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
        /bin/echo -e "${logmessage} OK"
    fi
    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`

    timespec
    /bin/echo -e "${STAMP} ${logmessage} OK" >> ${BOOTLOG}

    return 0
}

log_success_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"
    else
        echo " OK"
    fi

    echo " OK" >> ${BOOTLOG}

    return 0
}

#####
# log_failure_msg()                                                                                   #
# Usage: log_failure_msg ["message"]                                                                #
#                                                                                                     #
# Purpose: Print a failure status message to the screen and                                       #
#           a boot log file.                                                                        #
#                                                                                                     #
# Inputs:  $@ - Message                                                                              #
#                                                                                                     #
# Return values: Not used                                                                            #

```

```
#####
log_failure_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
        logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
        /bin/echo -e "${logmessage} FAIL"
    fi

    # Strip non-printable characters from log file

    timespec
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    /bin/echo -e "${STAMP} ${logmessage} FAIL" >> ${BOOTLOG}

    return 0
}

log_failure_msg2()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"
    else
        echo "FAIL"
    fi

    echo "FAIL" >> ${BOOTLOG}

    return 0
}

#####
# log_warning_msg() #
# Usage: log_warning_msg ["message"] #
# # #
# Purpose: Print a warning status message to the screen and #
# a boot log file. #
# # #
# Return values: Not used #
#####
log_warning_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
    then
        /bin/echo -n -e "${BMPREFIX}${@}"
        /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"
    else
        logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
        /bin/echo -e "${logmessage} WARN"
    fi

    # Strip non-printable characters from log file
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    timespec
    /bin/echo -e "${STAMP} ${logmessage} WARN" >> ${BOOTLOG}

    return 0
}

log_skip_msg()
{
    if [ "x$HEADLESS" != "xyes" ]
```

```

then
  /bin/echo -n -e "${BMPREFIX}${@}"
  /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"
else
  logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
  /bin/echo "SKIP"
fi

# Strip non-printable characters from log file
logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
/bin/echo "SKIP" >> ${BOOTLOG}

return 0
}

#####
# log_info_msg() #
# Usage: log_info_msg message #
# # #
# Purpose: Print an information message to the screen and #
# a boot log file. Does not print a trailing newline character. #
# # #
# Return values: Not used #
#####
log_info_msg()
{
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${BMPREFIX}${@}"
  else
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}"
  fi

  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
  timespec
  /bin/echo -n -e "${STAMP} ${logmessage}" >> ${BOOTLOG}

  return 0
}

log_info_msg2()
{
  if [ "x$HEADLESS" != "xyes" ]
  then
    /bin/echo -n -e "${@}"
  else
    logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
    /bin/echo -n -e "${logmessage}"
  fi

  # Strip non-printable characters from log file
  logmessage=`echo "${@}" | sed 's/\\033[^a-zA-Z]*.//g'`
  /bin/echo -n -e "${logmessage}" >> ${BOOTLOG}

  return 0
}

#####
# evaluate_retval() #
# Usage: Evaluate a return value and print success or failure as appropriate #
# # #
# Purpose: Convenience function to terminate an info message #
# # #
# Return values: Not used #
#####

```

```

evaluate_retval()
{
    local error_value="${?}"

    if [ ${error_value} = 0 ]; then
        log_success_msg2
    else
        log_failure_msg2
    fi
}

#####
# check_signal()
# Usage: check_signal [ -{signal} ]
#
# Purpose: Check for a valid signal. This is not defined by any LSB draft,
#         however, it is required to check the signals to determine if the
#         signals chosen are invalid arguments to the other functions.
#
# Inputs: Accepts a single string value in the form of -{signal}
#
# Return values:
#     0 - Success (signal is valid)
#     1 - Signal is not valid
#####
check_signal()
{
    local valsig

    # Add error handling for invalid signals
    valsig="-ALRM -HUP -INT -KILL -PIPE -POLL -PROF -TERM -USR1 -USR2"
    valsig="${valsig} -VTALRM -STKFLT -PWR -WINCH -CHLD -URG -TSTP -TTIN"
    valsig="${valsig} -TTOU -STOP -CONT -ABRT -FPE -ILL -QUIT -SEGV -TRAP"
    valsig="${valsig} -SYS -EMT -BUS -XCPU -XFSZ -0 -1 -2 -3 -4 -5 -6 -8 -9"
    valsig="${valsig} -11 -13 -14 -15 "

    echo "${valsig}" | grep -- " ${1} " > /dev/null

    if [ "${?}" -eq "0" ]; then
        return 0
    else
        return 1
    fi
}

#####
# check_sig_type()
# Usage: check_signal [ -{signal} | {signal} ]
#
# Purpose: Check if signal is a program termination signal or a control signal
#         This is not defined by any LSB draft, however, it is required to
#         check the signals to determine if they are intended to end a
#         program or simply to control it.
#
# Inputs: Accepts a single string value in the form or -{signal} or {signal}
#
# Return values:
#     0 - Signal is used for program termination
#     1 - Signal is used for program control
#####
check_sig_type()
{
    local valsig

    # The list of termination signals (limited to generally used items)
    valsig="-ALRM -INT -KILL -TERM -PWR -STOP -ABRT -QUIT -2 -3 -6 -9 -14 -15 "

```

```

echo "${valsig}" | grep -- " ${1} " > /dev/null

if [ "${?}" -eq "0" ]; then
    return 0
else
    return 1
fi
}

#####
# wait_for_user()
#
# Purpose: Wait for the user to respond if not a headless system
#
#####
wait_for_user()
{
    # Wait for the user by default
    [ "${HEADLESS=0}" = "0" ] && read ENTER
    return 0
}

#####
# is_true()
#
# Purpose: Utility to test if a variable is true | yes | 1
#
#####
is_true()
{
    [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ] || [ "$1" = "y" ] ||
    [ "$1" = "t" ]
}

# End /lib/lsb/init-functions

```

D.3. /etc/rc.d/init.d/mountvirtfs

```

#!/bin/sh
#####
# Begin mountvirtfs
#
# Description : Ensure proc, sysfs, run, and dev are mounted
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version      : LFS 12.0
#
#####

### BEGIN INIT INFO
# Provides:          mountvirtfs
# Required-Start:    $first
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Mounts various special fs needed at start
# Description:       Mounts /sys and /proc virtual (kernel) filesystems.
#                   Mounts /run (tmpfs) and /dev (devtmpfs).
#                   This is done only if they are not already mounted.
#                   with the kernel config proposed in the book, dev
#                   should be automatically mounted by the kernel.
#

```

```

# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
start)
    # Make sure /run is available before logging any messages
    if ! mountpoint /run >/dev/null; then
        mount /run || failed=1
    fi

    mkdir -p /run/lock
    chmod 1777 /run/lock

    log_info_msg "Mounting virtual file systems: ${INFO}/run"

    if ! mountpoint /proc >/dev/null; then
        log_info_msg2 " ${INFO}/proc"
        mount -o nosuid,noexec,nodev /proc || failed=1
    fi

    if ! mountpoint /sys >/dev/null; then
        log_info_msg2 " ${INFO}/sys"
        mount -o nosuid,noexec,nodev /sys || failed=1
    fi

    if ! mountpoint /dev >/dev/null; then
        log_info_msg2 " ${INFO}/dev"
        mount -o mode=0755,nosuid /dev || failed=1
    fi

    mkdir -p /dev/shm
    log_info_msg2 " ${INFO}/dev/shm"
    mount -o nosuid,nodev /dev/shm || failed=1

    mkdir -p /sys/fs/cgroup
    log_info_msg2 " ${INFO}/sys/fs/cgroup"
    mount -o nosuid,noexec,nodev /sys/fs/cgroup || failed=1

    (exit ${failed})
    evaluate_retval
    if [ "${failed}" = 1 ]; then
        exit 1
    fi

    log_info_msg "Create symlinks in /dev targeting /proc: ${INFO}/dev/stdin"
    ln -sf /proc/self/fd/0 /dev/stdin || failed=1

    log_info_msg2 " ${INFO}/dev/stdout"
    ln -sf /proc/self/fd/1 /dev/stdout || failed=1

    log_info_msg2 " ${INFO}/dev/stderr"
    ln -sf /proc/self/fd/2 /dev/stderr || failed=1

    log_info_msg2 " ${INFO}/dev/fd"
    ln -sf /proc/self/fd /dev/fd || failed=1

    if [ -e /proc/kcore ]; then
        log_info_msg2 " ${INFO}/dev/core"
        ln -sf /proc/kcore /dev/core || failed=1
    fi

    (exit ${failed})
    evaluate_retval
    exit $failed
;;

```

```

*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
esac

# End mountvirtfs

```

D.4. /etc/rc.d/init.d/modules

```

#!/bin/sh
#####
# Begin modules
#
# Description : Module auto-loading script
#
# Authors      : Zack Winkles
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      modules
# Required-Start: mountvirtfs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Loads required modules.
# Description:    Loads modules listed in /etc/sysconfig/modules.
# X-LFS-Provided-By: LFS
### END INIT INFO

# Assure that the kernel has module support.
[ -e /proc/modules ] || exit 0

. /lib/lsb/init-functions

case "${1}" in
    start)
        # Exit if there's no modules file or there are no
        # valid entries
        [ -r /etc/sysconfig/modules ] || exit 0
        grep -E -qv '^(#|)' /etc/sysconfig/modules || exit 0

        log_info_msg "Loading modules:"

        # Only try to load modules if the user has actually given us
        # some modules to load.

        while read module args; do

            # Ignore comments and blank lines.
            case "$module" in
                ""|"#"*) continue ;;
            esac

            # Attempt to load the module, passing any arguments provided.
            modprobe ${module} ${args} >/dev/null

            # Print the module name if successful, otherwise take note.
            if [ $? -eq 0 ]; then

```

```

        log_info_msg2 " ${module}"
    else
        failedmod="${failedmod} ${module}"
    fi
done < /etc/sysconfig/modules

# Print a message about successfully loaded modules on the correct line.
log_success_msg2

# Print a failure message with a list of any modules that
# may have failed to load.
if [ -n "${failedmod}" ]; then
    log_failure_msg "Failed to load modules:${failedmod}"
    exit 1
fi
;;

*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
esac

exit 0

# End modules

```

D.5. /etc/rc.d/init.d/udev

```

#!/bin/sh
#####
# Begin udev
#
# Description : Udev cold-plugging script
#
# Authors      : Zack Winkles, Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Xi Ruoyao - xry111@xry111.site
#
# Version      : LFS 12.0
#
#####

### BEGIN INIT INFO
# Provides:          udev $time
# Required-Start:    localnet
# Should-Start:      modules
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Populates /dev with device nodes.
# Description:        Mounts a tempfs on /dev and starts the udevd daemon.
#                     Device nodes are created as defined by udev.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Populating /dev with device nodes... "
        if ! grep -q '[:space:]sysfs' /proc/mounts; then
            log_failure_msg2
            msg="FAILURE:\n\nUnable to create "
            msg="${msg}devices without a SysFS filesystem\n\n"

```

```

    msg="{msg}After you press Enter, this system "
    msg="{msg}will be halted and powered off.\n\n"
    log_info_msg "$msg"
    log_info_msg "Press Enter to continue..."
    wait_for_user
    /etc/rc.d/init.d/halt start
fi

# Start the udev daemon to continually watch for, and act on,
# uevents
SYSTEMD_LOG_TARGET=kmsg /sbin/udev --daemon

# Now traverse /sys in order to "coldplug" devices that have
# already been discovered
/bin/udevadm trigger --action=add    --type=subsystems
/bin/udevadm trigger --action=add    --type=devices
/bin/udevadm trigger --action=change --type=devices

# Now wait for udevd to process the uevents we triggered
if ! is_true "$OMIT_UDEV_SETTLE"; then
    /bin/udevadm settle
fi

# If any LVM based partitions are on the system, ensure they
# are activated so they can be used.
if [ -x /sbin/vgchange ]; then /sbin/vgchange -a y >/dev/null; fi

log_success_msg2
;;

*)
    echo "Usage ${0} {start}"
    exit 1
;;

esac

exit 0

# End udev

```

D.6. /etc/rc.d/init.d/swap

```

#!/bin/sh
#####
# Begin swap
#
# Description : Swap Control Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          swap
# Required-Start:    udev
# Should-Start:      modules
# Required-Stop:     localnet
# Should-Stop:       $local_fs
# Default-Start:     S
# Default-Stop:      0 6
# Short-Description: Activates and deactivates swap partitions.
# Description:       Activates and deactivates swap partitions defined in
#                    /etc/fstab.
#

```

```

# X-LFS-Provided-By:   LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Activating all swap files/partitions..."
        swapon -a
        evaluate_retval
        ;;

    stop)
        log_info_msg "Deactivating all swap files/partitions..."
        swapoff -a
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        log_success_msg "Retrieving swap status."
        swapon -s
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

exit 0

# End swap

```

D.7. /etc/rc.d/init.d/setclock

```

#!/bin/sh
#####
# Begin setclock
#
# Description : Setting Linux Clock
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:
# Required-Start:
# Should-Start:      modules
# Required-Stop:
# Should-Stop:       $syslog
# Default-Start:     S
# Default-Stop:
# Short-Description: Stores and restores time from the hardware clock
# Description:        On boot, system time is obtained from hwclock. The
#                    hardware clock can also be set on shutdown.
#
# X-LFS-Provided-By: LFS

```

```

### END INIT INFO

. /lib/lsb/init-functions

[ -r /etc/sysconfig/clock ] && . /etc/sysconfig/clock

case "${UTC}" in
    yes|true|1)
        CLOCKPARAMS="${CLOCKPARAMS} --utc"
        ;;

    no|false|0)
        CLOCKPARAMS="${CLOCKPARAMS} --localtime"
        ;;

esac

case ${1} in
    start)
        hwclock --hctosys ${CLOCKPARAMS} >/dev/null
        ;;

    stop)
        log_info_msg "Setting hardware clock..."
        hwclock --systohc ${CLOCKPARAMS} >/dev/null
        evaluate_retval
        ;;

    *)
        echo "Usage: ${0} {start|stop}"
        exit 1
        ;;

esac

exit 0

```

D.8. /etc/rc.d/init.d/checkfs

```

#!/bin/sh
#####
# Begin checkfs
#
# Description : File System Check
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               A. Luebke - luebke@users.sourceforge.net
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Based on checkfs script from LFS-3.1 and earlier.
#
# From man fsck
# 0      - No errors
# 1      - File system errors corrected
# 2      - System should be rebooted
# 4      - File system errors left uncorrected
# 8      - Operational error
# 16     - Usage or syntax error
# 32     - Fsck canceled by user request
# 128    - Shared library error
#
#####

```

```

### BEGIN INIT INFO
# Provides:          checkfs
# Required-Start:    udev swap
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Checks local filesystems before mounting.
# Description:       Checks local filesystems before mounting.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
  start)
    if [ -f /fastboot ]; then
      msg="/fastboot found, will omit "
      msg="${msg} file system checks as requested.\n"
      log_info_msg "${msg}"
      exit 0
    fi

    log_info_msg "Mounting root file system in read-only mode... "
    mount -n -o remount,ro / >/dev/null

    if [ ${?} != 0 ]; then
      log_failure_msg2
      msg="\n\nCannot check root "
      msg="${msg}filesystem because it could not be mounted "
      msg="${msg}in read-only mode.\n\n"
      msg="${msg}After you press Enter, this system will be "
      msg="${msg}halted and powered off.\n\n"
      log_failure_msg "${msg}"

      log_info_msg "Press Enter to continue..."
      wait_for_user
      /etc/rc.d/init.d/halt start
    else
      log_success_msg2
    fi

    if [ -f /forcefsck ]; then
      msg="/forcefsck found, forcing file"
      msg="${msg} system checks as requested."
      log_success_msg "${msg}"
      options="-f"
    else
      options=""
    fi

    log_info_msg "Checking file systems..."
    # Note: -a option used to be -p; but this fails e.g. on fsck.minix
    if is_true "$VERBOSE_FSCK"; then
      fsck ${options} -a -A -C -T
    else
      fsck ${options} -a -A -C -T >/dev/null
    fi

    error_value=${?}

    if [ "${error_value}" = 0 ]; then
      log_success_msg2
    fi

    if [ "${error_value}" = 1 ]; then

```

```

msg="\nWARNING:\n\nFile system errors "
msg="${msg}were found and have been corrected.\n"
msg="${msg}      You may want to double-check that "
msg="${msg}everything was fixed properly."
log_warning_msg "$msg"
fi

if [ "${error_value}" = 2 -o "${error_value}" = 3 ]; then
msg="\nWARNING:\n\nFile system errors "
msg="${msg}were found and have been "
msg="${msg}corrected, but the nature of the "
msg="${msg}errors require this system to be rebooted.\n\n"
msg="${msg}After you press enter, "
msg="${msg}this system will be rebooted\n\n"
log_failure_msg "$msg"

log_info_msg "Press Enter to continue..."
wait_for_user
reboot -f
fi

if [ "${error_value}" -gt 3 -a "${error_value}" -lt 16 ]; then
msg="\nFAILURE:\n\nFile system errors "
msg="${msg}were encountered that could not be "
msg="${msg}fixed automatically.\nThis system "
msg="${msg}cannot continue to boot and will "
msg="${msg}therefore be halted until those "
msg="${msg}errors are fixed manually by a "
msg="${msg}System Administrator.\n\n"
msg="${msg}After you press Enter, this system will be "
msg="${msg}halted and powered off.\n\n"
log_failure_msg "$msg"

log_info_msg "Press Enter to continue..."
wait_for_user
/etc/rc.d/init.d/halt start
fi

if [ "${error_value}" -ge 16 ]; then
msg="FAILURE:\n\nUnexpected failure "
msg="${msg}running fsck. Exited with error "
msg="${msg} code: ${error_value}.\n"
log_info_msg $msg
exit ${error_value}
fi

exit 0
;;
*)
echo "Usage: ${0} {start}"
exit 1
;;
esac

# End checkfs

```

D.9. /etc/rc.d/init.d/mountfs

```

#!/bin/sh
#####
# Begin mountfs
#
# Description : File System Mount Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org

```

```

#
# Version      : LFS 7.0
#
#####
### BEGIN INIT INFO
# Provides:    $local_fs
# Required-Start:    udev checkfs
# Should-Start:    modules
# Required-Stop:    localnet
# Should-Stop:
# Default-Start:    S
# Default-Stop:    0 6
# Short-Description:    Mounts/unmounts local filesystems defined in /etc/fstab.
# Description:    Remounts root filesystem read/write and mounts all
#                  remaining local filesystems defined in /etc/fstab on
#                  start. Remounts root filesystem read-only and unmounts
#                  remaining filesystems on stop.
# X-LFS-Provided-By:    LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Remounting root file system in read-write mode..."
        mount --options remount,rw / >/dev/null
        evaluate_retval

        # Remove fsck-related file system watermarks.
        rm -f /fastboot /forcefsck

        # Make sure /dev/pts exists
        mkdir -p /dev/pts

        # This will mount all filesystems that do not have _netdev in
        # their option list. _netdev denotes a network filesystem.

        log_info_msg "Mounting remaining file systems..."
        failed=0
        mount --all --test-opts no_netdev >/dev/null || failed=1
        evaluate_retval
        exit $failed
        ;;

    stop)
        # Don't unmount virtual file systems like /run
        log_info_msg "Unmounting all other currently mounted file systems..."
        # Ensure any loop devices are removed
        losetup -D
        umount --all --detach-loop --read-only \
            --types notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
        evaluate_retval

        # Make sure / is mounted read only (umount bug)
        mount --options remount,ro /

        # Make all LVM volume groups unavailable, if appropriate
        # This fails if swap or / are on an LVM partition
        #if [ -x /sbin/vgchange ]; then /sbin/vgchange -an > /dev/null; fi
        if [ -r /etc/mdadm.conf ]; then
            log_info_msg "Mark arrays as clean..."
            mdadm --wait-clean --scan
            evaluate_retval
        fi
        ;;

```

```

*)
    echo "Usage: ${0} {start|stop}"
    exit 1
    ;;
esac

# End mountfs

```

D.10. /etc/rc.d/init.d/udev_retry

```

#!/bin/sh
#####
# Begin udev_retry
#
# Description : Udev cold-plugging script (retry)
#
# Authors      : Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Bryan Kadzban -
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      udev_retry
# Required-Start: udev
# Should-Start:  $local_fs cleanfs
# Required-Stop:
# Should-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Replays failed uevents and creates additional devices.
# Description:      Replays any failed uevents that were skipped due to
#                   slow hardware initialization, and creates those needed
#                   device nodes
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Retrying failed uevents, if any..."

        rundir=/run/udev
        # From Debian: "copy the rules generated before / was mounted
        # read-write":

        for file in ${rundir}/tmp-rules--*; do
            dest=${file##*tmp-rules--}
            [ "$dest" = '*' ] && break
            cat $file >> /etc/udev/rules.d/$dest
            rm -f $file
        done

        # Re-trigger the uevents that may have failed,
        # in hope they will succeed now
        /bin/sed -e 's/#.*$//' /etc/sysconfig/udev_retry | /bin/grep -v '^$' | \
        while read line ; do
            for subsystem in $line ; do
                /bin/udevadm trigger --subsystem-match=$subsystem --action=add
            done
        done

        # Now wait for udevd to process the uevents we triggered

```

```

    if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
        /bin/udevadm settle
    fi

    evaluate_retval
    ;;

*)
    echo "Usage ${0} {start}"
    exit 1
    ;;
esac

exit 0

# End udev_retry

```

D.11. /etc/rc.d/init.d/cleanfs

```

#!/bin/sh
#####
# Begin cleanfs
#
# Description : Clean file system
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      cleanfs
# Required-Start: $local_fs
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Cleans temporary directories early in the boot process.
# Description:     Cleans temporary directories /run, /var/lock, and
#                  optionally, /tmp. cleanfs also creates /run/utmp
#                  and any files defined in /etc/sysconfig/createfiles.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

# Function to create files/directory on boot.
create_files()
{
    # Input to file descriptor 9 and output to stdin (redirection)
    exec 9>&0 < /etc/sysconfig/createfiles

    while read name type perm usr grp dtype maj min junk
    do
        # Ignore comments and blank lines.
        case "${name}" in
            ""|\#*) continue ;;
        esac

        # Ignore existing files.
        if [ ! -e "${name}" ]; then
            # Create stuff based on its type.
            case "${type}" in

```

```

    dir)
        mkdir "${name}"
        ;;
    file)
        :> "${name}"
        ;;
    dev)
        case "${dtype}" in
            char)
                mknod "${name}" c ${maj} ${min}
                ;;
            block)
                mknod "${name}" b ${maj} ${min}
                ;;
            pipe)
                mknod "${name}" p
                ;;
            *)
                log_warning_msg "\nUnknown device type: ${dtype}"
                ;;
        esac
        ;;
    *)
        log_warning_msg "\nUnknown type: ${type}"
        continue
        ;;
esac

# Set up the permissions, too.
chown ${usr}:${grp} "${name}"
chmod ${perm} "${name}"
fi
done

# Close file descriptor 9 (end redirection)
exec 0>&9 9>&-
return 0
}

case "${1}" in
    start)
        log_info_msg "Cleaning file systems:"

        if [ "${SKIPTMPCLEAN}" = "" ]; then
            log_info_msg2 " /tmp"
            cd /tmp &&
            find . -xdev -mindepth 1 ! -name lost+found -delete || failed=1
        fi

        > /run/utmp

        if grep -q '^utmp:' /etc/group ; then
            chmod 664 /run/utmp
            chgrp utmp /run/utmp
        fi

        (exit ${failed})
        evaluate_retval

        if grep -E -qv '^(#|$)' /etc/sysconfig/createfiles 2>/dev/null; then
            log_info_msg "Creating files and directories... "
            create_files      # Always returns 0
            evaluate_retval
        fi

        exit $failed
    ;;

```

```

*)
    echo "Usage: ${0} {start}"
    exit 1
    ;;
esac

# End cleanfs

```

D.12. /etc/rc.d/init.d/console

```

#!/bin/sh
#####
# Begin console
#
# Description : Sets keymap and screen font
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               Alexander E. Patrakov
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:      console
# Required-Start: $local_fs
# Should-Start:  udev_retry
# Required-Stop:
# Should-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Sets up a localised console.
# Description:     Sets up fonts and language settings for the user's
#                  local as defined by /etc/sysconfig/console.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

# Native English speakers probably don't have /etc/sysconfig/console at all
[ -r /etc/sysconfig/console ] && . /etc/sysconfig/console

failed=0

case "${1}" in
    start)
        # See if we need to do anything
        if [ -z "${KEYMAP}" ] && [ -z "${KEYMAP_CORRECTIONS}" ] &&
            [ -z "${FONT}" ] && [ -z "${LEGACY_CHARSET}" ] &&
            ! is_true "${UNICODE}"; then
            exit 0
        fi

        # There should be no bogus failures below this line!
        log_info_msg "Setting up Linux console..."

        # Figure out if a framebuffer console is used
        [ -d /sys/class/graphics/fbcon ] && use_fb=1 || use_fb=0

        # Figure out the command to set the console into the
        # desired mode
        is_true "${UNICODE}" &&
            MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
            MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"
    esac

```

```

# On framebuffer consoles, font has to be set for each vt in
# UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

! is_true "${use_fb}" || [ -z "${FONT}" ] ||
  MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

# Apply that command to all consoles mentioned in
# /etc/inittab. Important: in the UTF-8 mode this should
# happen before setfont, otherwise a kernel bug will
# show up and the unicode map of the font will not be
# used.

for TTY in `grep '^[^#].*respawn:/sbin/agetty' /etc/inittab |
  grep -o '\btty[[:digit:]]*\b'`
do
  openvt -f -w -c ${TTY#tty} -- \
    /bin/sh -c "${MODE_COMMAND}" || failed=1
done

# Set the font (if not already set above) and the keymap
[ "${use_fb}" == "1" ] || [ -z "${FONT}" ] || setfont $FONT || failed=1

[ -z "${KEYMAP}" ] ||
  loadkeys ${KEYMAP} >/dev/null 2>&1 ||
  failed=1

[ -z "${KEYMAP_CORRECTIONS}" ] ||
  loadkeys ${KEYMAP_CORRECTIONS} >/dev/null 2>&1 ||
  failed=1

# Convert the keymap from $LEGACY_CHARSET to UTF-8
[ -z "${LEGACY_CHARSET}" ] ||
  dumpkeys -c "${LEGACY_CHARSET}" | loadkeys -u >/dev/null 2>&1 ||
  failed=1

# If any of the commands above failed, the trap at the
# top would set $failed to 1
( exit $failed )
evaluate_retval

exit $failed
;;

*)
  echo "Usage:  ${0} {start}"
  exit 1
  ;;
esac

# End console

```

D.13. /etc/rc.d/init.d/localnet

```

#!/bin/sh
#####
# Begin localnet
#
# Description : Loopback device
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

```

```

### BEGIN INIT INFO
# Provides:          localnet
# Required-Start:    mountvirtfs
# Should-Start:      modules
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:      0 6
# Short-Description: Starts the local network.
# Description:       Sets the hostname of the machine and starts the
#                   loopback interface.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions
[ -r /etc/sysconfig/network ] && . /etc/sysconfig/network
[ -r /etc/hostname ] && HOSTNAME=`cat /etc/hostname`

case "${1}" in
    start)
        log_info_msg "Bringing up the loopback interface..."
        ip addr add 127.0.0.1/8 label lo dev lo
        ip link set lo up
        evaluate_retval

        log_info_msg "Setting hostname to ${HOSTNAME}..."
        hostname ${HOSTNAME}
        evaluate_retval
        ;;

    stop)
        log_info_msg "Bringing down the loopback interface..."
        ip link set lo down
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        echo "Hostname is: $(hostname)"
        ip link show lo
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart|status}"
        exit 1
        ;;
esac

exit 0

# End localnet

```

D.14. /etc/rc.d/init.d/sysctl

```

#!/bin/sh
#####
# Begin sysctl
#
# Description : File uses /etc/sysctl.conf to set kernel runtime
#              parameters
#
# Authors      : Nathan Coulson (nathan@linuxfromscratch.org)

```

```

#           Matthew Burgess (matthew@linuxfromscratch.org)
#           DJ Lucas - dj@linuxfromscratch.org
# Update    : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version   : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          sysctl
# Required-Start:    mountvirtfs
# Should-Start:      console
# Required-Stop:
# Should-Stop:
# Default-Start:     S
# Default-Stop:
# Short-Description: Makes changes to the proc filesystem
# Description:        Makes changes to the proc filesystem as defined in
#                     /etc/sysctl.conf.  See 'man sysctl(8)'.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
  start)
    if [ -f "/etc/sysctl.conf" ]; then
      log_info_msg "Setting kernel runtime parameters..."
      sysctl -q -p
      evaluate_retval
    fi
    ;;

  status)
    sysctl -a
    ;;

  *)
    echo "Usage: ${0} {start|status}"
    exit 1
    ;;
esac

exit 0

# End sysctl

```

D.15. /etc/rc.d/init.d/sysklogd

```

#!/bin/sh
#####
# Begin sysklogd
#
# Description : Sysklogd loader
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org LFS12.1
#               Remove kernel log daemon.  The functionality has been
#               merged with syslogd.
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO

```

```

# Provides:          $syslog
# Required-Start:    $first localnet
# Should-Start:
# Required-Stop:     $local_fs
# Should-Stop:       sendsignals
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Starts system log daemon.
# Description:       Starts system log daemon.
#                   /etc/fstab.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Starting system log daemon..."
        parms=${SYSKLOGD_PARMS-'-m 0'}
        start_daemon /sbin/syslogd $parms
        evaluate_retval
        ;;

    stop)
        log_info_msg "Stopping system log daemon..."
        killproc /sbin/syslogd
        evaluate_retval
        ;;

    reload)
        log_info_msg "Reloading system log daemon config file..."
        pid=`pidofproc syslogd`
        kill -HUP "${pid}"
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    status)
        statusproc /sbin/syslogd
        ;;

    *)
        echo "Usage: ${0} {start|stop|reload|restart|status}"
        exit 1
        ;;
esac

exit 0

# End syslogd

```

D.16. /etc/rc.d/init.d/network

```

#!/bin/sh
#####
# Begin network
#
# Description : Network Control Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpfleming@linuxfromscratch.org

```

```

#           DJ Lucas - dj@linuxfromscratch.org
# Update    : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version   : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          $network
# Required-Start:    $local_fs localnet swap
# Should-Start:     $syslog firewalld iptables nftables
# Required-Stop:    $local_fs localnet swap
# Should-Stop:      $syslog firewalld iptables nftables
# Default-Start:    2 3 4 5
# Default-Stop:     0 1 6
# Short-Description: Starts and configures network interfaces.
# Description:      Starts and configures network interfaces.
# X-LFS-Provided-By: LFS
### END INIT INFO

case "${1}" in
    start)
        # if the default route exists, network is already configured
        if ip route | grep -q "^default"; then return 0; fi
        # Start all network interfaces
        for file in /etc/sysconfig/ifconfig.*
        do
            interface=${file##*/ifconfig.}

            # Skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]; then continue; fi

            /sbin/ifup ${interface}
        done
        ;;

    stop)
        # Unmount any network mounted file systems
        umount --all --force --types nfs,cifs,nfs4

        # Reverse list
        net_files=""
        for file in /etc/sysconfig/ifconfig.*
        do
            net_files="${file} ${net_files}"
        done

        # Stop all network interfaces
        for file in ${net_files}
        do
            interface=${file##*/ifconfig.}

            # Skip if $file is * (because nothing was found)
            if [ "${interface}" = "*" ]; then continue; fi

            # See if interface exists
            if [ ! -e /sys/class/net/${interface} ]; then continue; fi

            # Is interface UP?
            ip link show $interface 2>/dev/null | grep -q "state UP"
            if [ $? -ne 0 ]; then continue; fi

            /sbin/ifdown ${interface}
        done
        ;;

    restart)

```

```

    ${0} stop
    sleep 1
    ${0} start
    ;;

*)
    echo "Usage: ${0} {start|stop|restart}"
    exit 1
    ;;
esac

exit 0

# End network

```

D.17. /etc/rc.d/init.d/sendsignals

```

#!/bin/sh
#####
# Begin sendsignals
#
# Description : Sendsignals Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

### BEGIN INIT INFO
# Provides:          sendsignals
# Required-Start:
# Should-Start:
# Required-Stop:     $local_fs swap localnet
# Should-Stop:
# Default-Start:
# Default-Stop:      0 6
# Short-Description: Attempts to kill remaining processes.
# Description:       Attempts to kill remaining processes.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    stop)
        omit=$(pidof mdmon)
        [ -n "$omit" ] && omit="-o $omit"

        log_info_msg "Sending all processes the TERM signal..."
        killall15 -15 $omit
        error_value=${?}

        sleep ${KILLDELAY}

        if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
            log_success_msg
        else
            log_failure_msg
        fi

        log_info_msg "Sending all processes the KILL signal..."
        killall15 -9 $omit
        error_value=${?}
    ;;
)

```

```

    sleep ${KILLDELAY}

    if [ "${error_value}" = 0 -o "${error_value}" = 2 ]; then
        log_success_msg
    else
        log_failure_msg
    fi
    ;;

*)
    echo "Usage: ${0} {stop}"
    exit 1
    ;;

esac

exit 0

# End sendsignals

```

D.18. /etc/rc.d/init.d/reboot

```

#!/bin/sh
#####
# Begin reboot
#
# Description : Reboot Scripts
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Updates      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               Pierre Labastie - pierre@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
#####

### BEGIN INIT INFO
# Provides:          reboot
# Required-Start:    $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:     6
# Default-Stop:
# Short-Description: Reboots the system.
# Description:       Reboots the System.
# X-LFS-Provided-By: LFS
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Restarting system..."
        reboot -d -f -i
        ;;

    *)
        echo "Usage: ${0} {start}"
        exit 1
        ;;

esac

```

```
# End reboot
```

D.19. /etc/rc.d/init.d/halt

```
#!/bin/sh
#####
# Begin halt
#
# Description : Halt Script
#
# Authors      : Gerard Beekmans - gerard@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               : Pierre Labastie - pierre@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes        : Update March 24th, 2022: change "stop" to "start".
#               Add the $last facility to Required-start
#
#####

### BEGIN INIT INFO
# Provides:          halt
# Required-Start:    $last
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:    0
# Default-Stop:
# Short-Description: Halts the system.
# Description:       Halts the System.
# X-LFS-Provided-By: LFS
### END INIT INFO

case "${1}" in
    start)
        halt -d -f -i -p
        ;;

    *)
        echo "Usage: {start}"
        exit 1
        ;;
esac

# End halt
```

D.20. /etc/rc.d/init.d/template

```
#!/bin/sh
#####
# Begin scriptname
#
# Description :
#
# Authors      :
#
# Version      : LFS x.x
#
# Notes        :
#
#####
```

```

### BEGIN INIT INFO
# Provides:          template
# Required-Start:
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:
# Default-Stop:
# Short-Description:
# Description:
# X-LFS-Provided-By:
### END INIT INFO

. /lib/lsb/init-functions

case "${1}" in
    start)
        log_info_msg "Starting..."
        # if it is possible to use start_daemon
        start_daemon fully_qualified_path
        # if it is not possible to use start_daemon
        # (command to start the daemon is not simple enough)
        if ! pidofproc daemon_name_as_reported_by_ps >/dev/null; then
            command_to_start_the_service
        fi
        evaluate_retval
        ;;

    stop)
        log_info_msg "Stopping..."
        # if it is possible to use killproc
        killproc fully_qualified_path
        # if it is not possible to use killproc
        # (the daemon shouldn't be stopped by killing it)
        if pidofproc daemon_name_as_reported_by_ps >/dev/null; then
            command_to_stop_the_service
        fi
        evaluate_retval
        ;;

    restart)
        ${0} stop
        sleep 1
        ${0} start
        ;;

    *)
        echo "Usage: ${0} {start|stop|restart}"
        exit 1
        ;;
esac

exit 0

# End scriptname

```

D.21. /etc/sysconfig/modules

```

#####
# Begin /etc/sysconfig/modules
#
# Description : Module auto-loading configuration
#
# Authors      :
#
# Version      : 00.00
#

```

```
# Notes      : The syntax of this file is as follows:
#             <module> [<arg1> <arg2> ...]
#
# Each module should be on its own line, and any options that you want
# passed to the module should follow it.  The line delimitator is either
# a space or a tab.
#####

# End /etc/sysconfig/modules
```

D.22. /etc/sysconfig/createfiles

```
#####
# Begin /etc/sysconfig/createfiles
#
# Description : Createfiles script config file
#
# Authors      :
#
# Version      : 00.00
#
# Notes       : The syntax of this file is as follows:
#               if type is equal to "file" or "dir"
#                 <filename> <type> <permissions> <user> <group>
#               if type is equal to "dev"
#                 <filename> <type> <permissions> <user> <group> <devtype>
#                 <major> <minor>
#
#               <filename> is the name of the file which is to be created
#               <type> is either file, dir, or dev.
#                   file creates a new file
#                   dir creates a new directory
#                   dev creates a new device
#               <devtype> is either block, char or pipe
#                   block creates a block device
#                   char creates a character device
#                   pipe creates a pipe, this will ignore the <major> and
#                   <minor> fields
#               <major> and <minor> are the major and minor numbers used for
#               the device.
#####

# End /etc/sysconfig/createfiles
```

D.23. /etc/sysconfig/udev-retry

```
#####
# Begin /etc/sysconfig/udev_retry
#
# Description : udev_retry script configuration
#
# Authors      :
#
# Version      : 00.00
#
# Notes       : Each subsystem that may need to be re-triggered after mountfs
#               runs should be listed in this file.  Probable subsystems to be
#               listed here are rtc (due to /var/lib/hwclock/adjtime) and sound
#               (due to both /var/lib/alsa/asound.state and /usr/sbin/alsactl).
#               Entries are whitespace-separated.
#####

rtc

# End /etc/sysconfig/udev_retry
```

D.24. /sbin/ifup

```
#!/bin/sh
#####
# Begin /sbin/ifup
#
# Description : Interface Up
#
# Authors      : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpflaming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#               DJ Lucas - dj@linuxfromscratch.org
#
# Version     : LFS 7.7
#
# Notes      : The IFCONFIG variable is passed to the SERVICE script
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
#####

up()
{
    log_info_msg "Bringing up the ${1} interface..."

    if ip link show $1 > /dev/null 2>&1; then
        link_status=`ip link show $1`

        if [ -n "${link_status}" ]; then
            if ! echo "${link_status}" | grep -q UP; then
                ip link set $1 up
            fi
        fi

    else
        log_failure_msg "Interface ${IFACE} doesn't exist."
        exit 1
    fi

    evaluate_retval
}

RELEASE="7.7"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifup, version ${RELEASE}"

while [ $# -gt 0 ]; do
    case "$1" in
        --help | -h)      help="y"; break ;;

        --version | -V)   echo "${VERSTR}"; exit 0 ;;

        -*)               echo "ifup: ${1}: invalid option" >&2
                           echo "${USAGE}" >& 2
                           exit 2 ;;

        *)               break ;;
    esac
done

if [ -n "$help" ]; then
    echo "${VERSTR}"
    echo "${USAGE}"
    echo
    cat << HERE_EOF
ifup is used to bring up a network interface. The interface
```

parameter, e.g. eth0 or eth0:2, must match the trailing part of the interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

```

HERE_EOF
    exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%*"~"}" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
    log_failure_msg "Unable to bring up ${1} interface! ${file} is missing or cannot be accessed."
    exit 1
fi

. $file

if [ "$IFACE" = "" ]; then
    log_failure_msg "Unable to bring up ${1} interface! ${file} does not define an interface [IFACE]."
    exit 1
fi

# Do not process this service if started by boot, and ONBOOT
# is not set to yes
if [ "${IN_BOOT}" = "1" -a "${ONBOOT}" != "yes" ]; then
    exit 0
fi

# Bring up the interface
if [ "$VIRTINT" != "yes" ]; then
    up ${IFACE}
fi

for S in ${SERVICE}; do
    if [ ! -x "/lib/services/${S}" ]; then
        MSG="\nUnable to process ${file}. Either "
        MSG="${MSG}the SERVICE '${S}' was not present "
        MSG="${MSG}or cannot be executed."
        log_failure_msg "$MSG"
        exit 1
    fi
done

#if [ "${SERVICE}" = "wpa" ]; then log_success_msg; fi

# Create/configure the interface
for S in ${SERVICE}; do
    IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Set link up virtual interfaces
if [ "$VIRTINT" == "yes" ]; then
    up ${IFACE}
fi

# Bring up any additional interface components
for I in $INTERFACE_COMPONENTS; do up $I; done

# Set MTU if requested. Check if MTU has a "good" value.
if test -n "${MTU}"; then
    if [[ ${MTU} =~ ^[0-9]+$ ]] && [[ $MTU -ge 68 ]]; then
        for I in $IFACE $INTERFACE_COMPONENTS; do
            ip link set dev $I mtu $MTU;
        done
    fi
fi

```

```

done
else
    log_info_msg2 "Invalid MTU $MTU"
fi
fi

# Set the route default gateway if requested
if [ -n "${GATEWAY}" ]; then
    if ip route | grep -q default; then
        log_warning_msg "Gateway already setup; skipping."
    else
        log_info_msg "Adding default gateway ${GATEWAY} to the ${IFACE} interface..."
        ip route add default via ${GATEWAY} dev ${IFACE}
        evaluate_retval
    fi
fi

# End /sbin/ifup

```

D.25. /sbin/ifdown

```

#!/bin/bash
#####
# Begin /sbin/ifdown
#
# Description : Interface Down
#
# Authors      : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kpflaming@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
# Notes       : the IFCONFIG variable is passed to the scripts found
#               in the /lib/services directory, to indicate what file the
#               service should source to get interface specifications.
#
#####
RELEASE="7.0"

USAGE="Usage: $0 [ -hV ] [--help] [--version] interface"
VERSTR="LFS ifdown, version ${RELEASE}"

while [ $# -gt 0 ]; do
    case "$1" in
        --help | -h)      help="y"; break ;;

        --version | -V)   echo "${VERSTR}"; exit 0 ;;

        -*)               echo "ifup: ${1}: invalid option" >&2
                          echo "${USAGE}" >& 2
                          exit 2 ;;

        *)               break ;;
    esac
done

if [ -n "$help" ]; then
    echo "${VERSTR}"
    echo "${USAGE}"
    echo
    cat << HERE_EOF
ifdown is used to bring down a network interface. The interface
parameter, e.g. eth0 or eth0:2, must match the trailing part of the
interface specifications file, e.g. /etc/sysconfig/ifconfig.eth0:2.

```

```

HERE_EOF
    exit 0
fi

file=/etc/sysconfig/ifconfig.${1}

# Skip backup files
[ "${file}" = "${file%}"~" ] || exit 0

. /lib/lsb/init-functions

if [ ! -r "${file}" ]; then
    log_warning_msg "${file} is missing or cannot be accessed."
    exit 1
fi

. ${file}

if [ "$IFACE" = "" ]; then
    log_failure_msg "${file} does not define an interface [IFACE]."
    exit 1
fi

# We only need to first service to bring down the interface
S=`echo ${SERVICE} | cut -f1 -d" "`

if ip link show ${IFACE} > /dev/null 2>&1; then
    if [ -n "${S}" -a -x "/lib/services/${S}" ]; then
        IFCONFIG=${file} /lib/services/${S} ${IFACE} down
    else
        MSG="Unable to process ${file}. Either "
        MSG="${MSG}the SERVICE variable was not set "
        MSG="${MSG}or the specified service cannot be executed."
        log_failure_msg "$MSG"
        exit 1
    fi
else
    log_warning_msg "Interface ${1} doesn't exist."
fi

# Leave the interface up if there are additional interfaces in the device
link_status=`ip link show ${IFACE} 2>/dev/null`

if [ -n "${link_status}" ]; then
    if [ "$(echo "${link_status}" | grep UP)" != "" ]; then
        if [ "$(ip addr show ${IFACE} | grep 'inet ')" == "" ]; then
            log_info_msg "Bringing down the ${IFACE} interface..."
            ip link set ${IFACE} down
            evaluate_retval
        fi
    fi
fi

# End /sbin/ifdown

```

D.26. /lib/services/ipv4-static

```

#!/bin/sh
#####
# Begin /lib/services/ipv4-static
#
# Description : IPV4 Static Boot Script
#
# Authors      : Nathan Coulson - nathan@linuxfromscratch.org
#               Kevin P. Fleming - kp Fleming@linuxfromscratch.org
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#

```

```

# Version      : LFS 7.0
#
#####

. /lib/lsb/init-functions
. ${IFCONFIG}

if [ -z "${IP}" ]; then
    log_failure_msg "\nIP variable missing from ${IFCONFIG}, cannot continue."
    exit 1
fi

if [ -z "${PREFIX}" -a -z "${PEER}" ]; then
    log_warning_msg "\nPREFIX variable missing from ${IFCONFIG}, assuming 24."
    PREFIX=24
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PREFIX}" -a -n "${PEER}" ]; then
    log_failure_msg "\nPREFIX and PEER both specified in ${IFCONFIG}, cannot continue."
    exit 1
elif [ -n "${PREFIX}" ]; then
    args="${args} ${IP}/${PREFIX}"
elif [ -n "${PEER}" ]; then
    args="${args} ${IP} peer ${PEER}"
fi

if [ -n "${LABEL}" ]; then
    args="${args} label ${LABEL}"
fi

if [ -n "${BROADCAST}" ]; then
    args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
    up)
        if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then
            log_info_msg "Adding IPv4 address ${IP} to the ${1} interface..."
            ip addr add ${args} dev ${1}
            evaluate_retval
        else
            log_warning_msg "Cannot add IPv4 address ${IP} to ${1}.  Already present."
        fi
        ;;

    down)
        if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" != "" ]; then
            log_info_msg "Removing IPv4 address ${IP} from the ${1} interface..."
            ip addr del ${args} dev ${1}
            evaluate_retval
        fi

        if [ -n "${GATEWAY}" ]; then
            # Only remove the gateway if there are no remaining ipv4 addresses
            if [ "$(ip addr show ${1} 2>/dev/null | grep 'inet ')" != "" ]; then
                log_info_msg "Removing default gateway..."
                ip route del default
                evaluate_retval
            fi
        fi
        ;;

    *)
        echo "Usage: ${0} [interface] {up|down}"
        exit 1

```

```
;;
esac

# End /lib/services/ipv4-static
```

D.27. /lib/services/ipv4-static-route

```
#!/bin/sh
#####
# Begin /lib/services/ipv4-static-route
#
# Description : IPV4 Static Route Script
#
# Authors      : Kevin P. Fleming - kpfleming@linuxfromscratch.org
#              : DJ Lucas - dj@linuxfromscratch.org
# Update       : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version      : LFS 7.0
#
#####

. /lib/lsb/init-functions
. ${IFCONFIG}

case "${TYPE}" in
    (" | "network")
        need_ip=1
        need_gateway=1
        ;;

    ("default")
        need_gateway=1
        args="${args} default"
        desc="default"
        ;;

    ("host")
        need_ip=1
        ;;

    ("unreachable")
        need_ip=1
        args="${args} unreachable"
        desc="unreachable "
        ;;

    (*)
        log_failure_msg "Unknown route type (${TYPE}) in ${IFCONFIG}, cannot continue."
        exit 1
        ;;
esac

if [ -n "${GATEWAY}" ]; then
    MSG="The GATEWAY variable cannot be set in ${IFCONFIG} for static routes.\n"
    log_failure_msg "$MSG Use STATIC_GATEWAY only, cannot continue"
    exit 1
fi

if [ -n "${need_ip}" ]; then
    if [ -z "${IP}" ]; then
        log_failure_msg "IP variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi

    if [ -z "${PREFIX}" ]; then
        log_failure_msg "PREFIX variable missing from ${IFCONFIG}, cannot continue."
        exit 1
    fi
fi
```

```

fi

args="${args} ${IP}/${PREFIX}"
desc="${desc}${IP}/${PREFIX}"
fi

if [ -n "${need_gateway}" ]; then
  if [ -z "${STATIC_GATEWAY}" ]; then
    log_failure_msg "STATIC_GATEWAY variable missing from ${IFCONFIG}, cannot continue."
    exit 1
  fi
  args="${args} via ${STATIC_GATEWAY}"
fi

if [ -n "${SOURCE}" ]; then
  args="${args} src ${SOURCE}"
fi

case "${2}" in
  up)
    log_info_msg "Adding '${desc}' route to the ${1} interface..."
    ip route add ${args} dev ${1}
    evaluate_retval
    ;;
  down)
    log_info_msg "Removing '${desc}' route from the ${1} interface..."
    ip route del ${args} dev ${1}
    evaluate_retval
    ;;
  *)
    echo "Usage: ${0} [interface] {up|down}"
    exit 1
    ;;
esac

# End /lib/services/ipv4-static-route

```

Apêndice E. Regras de configuração do Udev

As regras neste anexo estão listadas por conveniência. A instalação normalmente é feita via instruções na Seção 8.76, “Udev originário de Systemd-256.4.”

E.1. 55-lfs.rules

```
# /etc/udev/rules.d/55-lfs.rules: Rule definitions for LFS.

# Core kernel devices

# This causes the system clock to be set as soon as /dev/rtc becomes available.
SUBSYSTEM=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
KERNEL=="rtc", ACTION=="add", MODE="0644", RUN+="/etc/rc.d/init.d/setclock start"
```

Apêndice F. Licenças do LFS

Este livro está licenciado sob a Licença Creative Commons Atribuição-Use Não Comercial-Compartilhamento pela mesma licença 2.0.

As instruções de computador podem ser extraídas a partir do livro sob a Licença do MIT.

F.1. Licença da Creative Commons

Código Jurídico da Creative Commons

Atribuição - Uso 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/>.

F.2. A Licença do MIT

Direitos autorais © 1999-2024 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: 136
- Attr: 135
- Autoconf: 173
- Automake: 174
- Bash: 159
 - ferramentas: 62
- Bash: 159
 - ferramentas: 62
- Bc: 120
- Binutils: 128
 - ferramentas, passagem 1: 47
 - ferramentas, passagem 2: 75
- Binutils: 128
 - ferramentas, passagem 1: 47
 - ferramentas, passagem 2: 75
- Binutils: 128
 - ferramentas, passagem 1: 47
 - ferramentas, passagem 2: 75
- Bison: 157
 - ferramentas: 85
- Bison: 157
 - ferramentas: 85
- Scripts de inicialização: 241
 - uso: 252
- Scripts de inicialização: 241
 - uso: 252
- Bzip2: 110
- Check: 193
- Coreutils: 188
 - ferramentas: 63
- Coreutils: 188
 - ferramentas: 63
- DejaGNU: 126
- Diffutils: 194
 - ferramentas: 64
- Diffutils: 194
 - ferramentas: 64
- E2fsprogs: 232
- Expat: 164
- Expect: 124
- File: 116
 - ferramentas: 65
- File: 116
 - ferramentas: 65
- Findutils: 196
 - ferramentas: 66
- Findutils: 196
 - ferramentas: 66
- Flex: 121
- Flit-core: 183
- Gawk: 195
 - ferramentas: 67
- Gawk: 195
 - ferramentas: 67
- GCC: 144
 - ferramentas, libstdc++ passagem 1: 56
 - ferramentas, passagem 1: 49
 - ferramentas, passagem 2: 76
- GCC: 144
 - ferramentas, libstdc++ passagem 1: 56
 - ferramentas, passagem 1: 49
 - ferramentas, passagem 2: 76
- GCC: 144
 - ferramentas, libstdc++ passagem 1: 56
 - ferramentas, passagem 1: 49
 - ferramentas, passagem 2: 76
- GCC: 144
 - ferramentas, libstdc++ passagem 1: 56
 - ferramentas, passagem 1: 49
 - ferramentas, passagem 2: 76
- GDBM: 162
- Gettext: 155
 - ferramentas: 84
- Gettext: 155
 - ferramentas: 84
- Glibc: 101
 - ferramentas: 53
- Glibc: 101
 - ferramentas: 53
- GMP: 131
- Gperf: 163
- Grep: 158
 - ferramentas: 68
- Grep: 158
 - ferramentas: 68
- Groff: 197
- GRUB: 200
- Gzip: 202
 - ferramentas: 69
- Gzip: 202
 - ferramentas: 69
- Iana-Etc: 100
- Inetutils: 165
- Intltool: 172
- IPRoute2: 203
- Jinja2: 218
- Kbd: 205

Kmod: 177
 Less: 167
 Libcap: 137
 Libelf: 179
 libffi: 180
 Libpipeline: 208
 Libtool: 161
 Libxcrypt: 138
 Linux: 267
 ferramentas, cabeçalhos da API: 52
 Linux: 267
 ferramentas, cabeçalhos da API: 52
 Lz4: 114
 M4: 119
 ferramentas: 59
 M4: 119
 ferramentas: 59
 Make: 209
 ferramentas: 70
 Make: 209
 ferramentas: 70
 Man-DB: 222
 Páginas-Manual: 99
 MarkupSafe: 217
 Meson: 187
 MPC: 134
 MPFR: 133
 Ncurses: 150
 ferramentas: 60
 Ncurses: 150
 ferramentas: 60
 Ninja: 186
 OpenSSL: 175
 Patch: 210
 ferramentas: 71
 Patch: 210
 ferramentas: 71
 Perl: 168
 ferramentas: 86
 Perl: 168
 ferramentas: 86
 Pkgconf: 127
 Procps-ng: 225
 Psmisc: 154
 Python: 181
 temporário: 87
 Python: 181
 temporário: 87
 rc.site: 258
 Readline: 117
 Sed: 153
 ferramentas: 72
 Sed: 153
 ferramentas: 72
 Setuptools: 185
 Shadow: 140
 configurando: 141
 Shadow: 140
 configurando: 141
 Syslogd: 235
 configurando: 235
 Syslogd: 235
 configurando: 235
 SysVinit: 236
 configurando: 253
 SysVinit: 236
 configurando: 253
 Tar: 211
 ferramentas: 73
 Tar: 211
 ferramentas: 73
 Tcl: 122
 Texinfo: 212
 temporário: 88
 Texinfo: 212
 temporário: 88
 Udev: 219
 configurando: 220
 uso: 243
 Udev: 219
 configurando: 220
 uso: 243
 Udev: 219
 configurando: 220
 uso: 243
 Util-linux: 227
 ferramentas: 89
 Util-linux: 227
 ferramentas: 89
 Vim: 214
 wheel: 184
 XML::Parser: 171
 Xz: 112
 ferramentas: 74
 Xz: 112
 ferramentas: 74
 Zlib: 109
 zstd: 115

[: 188, 189

2to3: 181
 accessdb: 222, 223
 aclocal: 174, 174
 aclocal-1.17: 174, 174
 addftinfo: 197, 197
 addpart: 227, 228
 addr2line: 128, 129
 afmtodit: 197, 197
 agetty: 227, 228
 apropos: 222, 223
 ar: 128, 129
 as: 128, 129
 attr: 135, 135
 autoconf: 173, 173
 autoheader: 173, 173
 autom4te: 173, 173
 automake: 174, 174
 automake-1.17: 174, 174
 autopoint: 155, 155
 autoreconf: 173, 173
 autoscan: 173, 173
 autoupdate: 173, 173
 awk: 195, 195
 b2sum: 188, 189
 badblocks: 232, 233
 base64: 188, 189, 188, 189
 base64: 188, 189, 188, 189
 basename: 188, 189
 basenc: 188, 189
 bash: 159, 160
 bashbug: 159, 160
 bc: 120, 120
 bison: 157, 157
 blkdiscard: 227, 228
 blkid: 227, 228
 blkzone: 227, 228
 blockdev: 227, 228
 bomtool: 127, 127
 bootlogd: 236, 236
 bridge: 203, 203
 bunzip2: 110, 111
 bzcat: 110, 111
 bzcmp: 110, 111
 bzdiff: 110, 111
 bzegrep: 110, 111
 bzfgrep: 110, 111
 bzgrep: 110, 111
 bzip2: 110, 111
 bzip2recover: 110, 111
 bzless: 110, 111
 bzmored: 110, 111
 c++: 144, 148
 c++filt: 128, 129
 cal: 227, 228
 capsh: 137, 137
 captinfo: 150, 151
 cat: 188, 189
 catman: 222, 223
 cc: 144, 148
 cfdisk: 227, 228
 chacl: 136, 136
 chage: 140, 142
 chatr: 232, 233
 chcon: 188, 189
 chcpu: 227, 228
 checkmk: 193, 193
 chem: 197, 197
 chfn: 140, 142
 chpasswd: 140, 142
 chgrp: 188, 189
 chmem: 227, 228
 chmod: 188, 189
 choom: 227, 228
 chown: 188, 189
 chpasswd: 140, 142
 chroot: 188, 190
 chrt: 227, 228
 chsh: 140, 142
 chvt: 205, 206
 cksum: 188, 190
 clear: 150, 151
 cmp: 194, 194
 col: 227, 228
 colcrt: 227, 228
 colrm: 227, 228
 column: 227, 228
 comm: 188, 190
 compile_et: 232, 233
 corelist: 168, 169
 cp: 188, 190
 cpan: 168, 169
 cpp: 144, 148
 csplit: 188, 190
 ctrlaltdel: 227, 228
 ctstat: 203, 203
 cut: 188, 190
 c_rehash: 175, 176
 date: 188, 190
 dc: 120, 120
 dd: 188, 190

deallocvt: 205, 206
 debugfs: 232, 233
 dejagnu: 126, 126
 delpart: 227, 228
 depmod: 177, 177
 df: 188, 190
 diff: 194, 194
 diff3: 194, 194
 dir: 188, 190
 dircolors: 188, 190
 dirname: 188, 190
 dmesg: 227, 228
 dnsdomainname: 165, 166
 du: 188, 190
 dumpe2fs: 232, 233
 dumpkeys: 205, 206
 e2freefrag: 232, 233
 e2fsck: 232, 233
 e2image: 232, 233
 e2label: 232, 233
 e2mmpstatus: 232, 233
 e2scrub: 232, 233
 e2scrub_all: 232, 233
 e2undo: 232, 233
 e4crypt: 232, 233
 e4defrag: 232, 233
 echo: 188, 190
 egrep: 158, 158
 eject: 227, 229
 elfedit: 128, 129
 enc2xs: 168, 169
 encguess: 168, 169
 env: 188, 190
 envsubst: 155, 155
 eqn: 197, 197
 eqn2graph: 197, 197
 ex: 214, 216
 expand: 188, 190
 expect: 124, 125
 expiry: 140, 142
 expr: 188, 190
 factor: 188, 190
 faillog: 140, 142
 fallocate: 227, 229
 false: 188, 190
 fdisk: 227, 229
 fgconsole: 205, 206
 fgrep: 158, 158
 file: 116, 116
 filefrag: 232, 233
 findcore: 227, 229
 find: 196, 196
 findfs: 227, 229
 findmnt: 227, 229
 flex: 121, 121
 flex++: 121, 121
 flock: 227, 229
 fmt: 188, 190
 fold: 188, 190
 free: 225, 225
 fsck: 227, 229
 fsck.cramfs: 227, 229
 fsck.ext2: 232, 233
 fsck.ext3: 232, 233
 fsck.ext4: 232, 234
 fsck.minix: 227, 229
 fsfreeze: 227, 229
 fstab-decode: 236, 236
 fstrim: 227, 229
 ftp: 165, 166
 fuser: 154, 154
 g++: 144, 148
 gawk: 195, 195
 gawk-5.3.0: 195, 195
 gcc: 144, 148
 gc-ar: 144, 148
 gc-nm: 144, 148
 gc-ranlib: 144, 148
 gcov: 144, 148
 gcov-dump: 144, 148
 gcov-tool: 144, 148
 gdbmtool: 162, 162
 gdbm_dump: 162, 162
 gdbm_load: 162, 162
 gdiffmk: 197, 197
 gencat: 101, 107
 genl: 203, 203
 getcap: 137, 137
 getconf: 101, 107
 getent: 101, 107
 getfacl: 136, 136
 getfattr: 135, 135
 getkeycodes: 205, 206
 getopt: 227, 229
 getpcaps: 137, 137
 getsubids: 140, 142
 gettext: 155, 155
 gettext.sh: 155, 155
 gettextize: 155, 155
 glilypond: 197, 197

gpasswd: 140, 142
 gperf: 163, 163
 gperl: 197, 197
 gpinyin: 197, 197
 gprof: 128, 129
 gprofng: 128, 129
 grap2graph: 197, 198
 grep: 158, 158
 grn: 197, 198
 grodvi: 197, 198
 groff: 197, 198
 groffer: 197, 198
 grog: 197, 198
 grolbp: 197, 198
 grolj4: 197, 198
 gropdf: 197, 198
 groups: 197, 198
 grotty: 197, 198
 groupadd: 140, 142
 groupdel: 140, 142
 groupmems: 140, 142
 groupmod: 140, 143
 groups: 188, 190
 grpck: 140, 143
 grpconv: 140, 143
 grpunconv: 140, 143
 grub-bios-setup: 200, 201
 grub-editenv: 200, 201
 grub-file: 200, 201
 grub-fstest: 200, 201
 grub-glue-efi: 200, 201
 grub-install: 200, 201
 grub-kbdcomp: 200, 201
 grub-macbless: 200, 201
 grub-menulst2cfg: 200, 201
 grub-mkconfig: 200, 201
 grub-mkimage: 200, 201
 grub-mklayout: 200, 201
 grub-mknetdir: 200, 201
 grub-mkpasswd-pbkdf2: 200, 201
 grub-mkreldir: 200, 201
 grub-mkrescue: 200, 201
 grub-mkstandalone: 200, 201
 grub-ofpathname: 200, 201
 grub-probe: 200, 201
 grub-reboot: 200, 201
 grub-render-label: 200, 201
 grub-script-check: 200, 201
 grub-set-default: 200, 201
 grub-setup: 200, 201
 grub-syslinux2cfg: 200, 201
 gunzip: 202, 202
 gzexe: 202, 202
 gzip: 202, 202
 h2ph: 168, 169
 h2xs: 168, 169
 parar: 236, 236
 hardlink: 227, 229
 head: 188, 190
 hexdump: 227, 229
 hostid: 188, 190
 hostname: 165, 166
 hpftodit: 197, 198
 hwclock: 227, 229
 i386: 227, 229
 iconv: 101, 107
 iconvconfig: 101, 107
 id: 188, 190
 idle3: 181
 ifconfig: 165, 166
 ifnames: 173, 173
 ifstat: 203, 203
 indxbib: 197, 198
 info: 212, 212
 infocmp: 150, 151
 infotocap: 150, 151
 init: 236, 236
 insmod: 177, 178
 install: 188, 190
 install-info: 212, 213
 instmodsh: 168, 169
 intltool-extract: 172, 172
 intltool-merge: 172, 172
 intltool-prepare: 172, 172
 intltool-update: 172, 172
 intltoolize: 172, 172
 ionice: 227, 229
 ip: 203, 203
 ipcmk: 227, 229
 ipcrm: 227, 229
 ipcs: 227, 229
 irqtop: 227, 229
 isosize: 227, 229
 join: 188, 190
 json_pp: 168, 169
 kbdfinfo: 205, 206
 kbdrate: 205, 206
 kbd_mode: 205, 206
 kill: 227, 229
 killall: 154, 154

killall5: 236, 236
 kmod: 177, 178
 last: 227, 229
 lastb: 227, 229
 ld: 128, 129
 ld.bfd: 128, 129
 ld.gold: 128, 129
 ldattach: 227, 229
 ldconfig: 101, 107
 ldd: 101, 107
 lddlibc4: 101, 107
 less: 167, 167
 lessecho: 167, 167
 lesskey: 167, 167
 lex: 121, 121
 lexgrog: 222, 223
 lfskernel-6.10.5: 267, 272
 libasan: 144, 148
 libatomic: 144, 148
 libcc1: 144, 148
 libnetcfg: 168, 169
 libtool: 161, 161
 libtoolize: 161, 161
 link: 188, 190
 linux32: 227, 229
 linux64: 227, 229
 lkbib: 197, 198
 ln: 188, 190
 lstat: 203, 204
 loadkeys: 205, 206
 loadunimap: 205, 206
 locale: 101, 107
 localedef: 101, 107
 locate: 196, 196
 logger: 227, 229
 login: 140, 143
 logname: 188, 190
 logoutd: 140, 143
 logsave: 232, 234
 look: 227, 229
 lookbib: 197, 198
 losetup: 227, 229
 ls: 188, 190
 lsattr: 232, 234
 lsblk: 227, 229
 lscpu: 227, 229
 lsfd: 227, 230
 lspic: 227, 230
 lsirq: 227, 230
 lslocks: 227, 230
 lslogins: 227, 230
 lsmem: 227, 230
 lsmod: 177, 178
 lsns: 227, 230
 lto-dump: 144, 148
 lz4: 114, 114
 lz4c: 114, 114
 lz4cat: 112, 114
 lzcat: 112, 112
 lzcmp: 112, 112
 lzdiff: 112, 112
 lzegrep: 112, 112
 lzfgrep: 112, 112
 lzgrep: 112, 112
 lzless: 112, 112
 lzma: 112, 112
 lzmadec: 112, 112
 lzmainfo: 112, 112
 lzmore: 112, 112
 m4: 119, 119
 make: 209, 209
 makedb: 101, 107
 makeinfo: 212, 213
 man: 222, 224
 man-recode: 222, 224
 mandb: 222, 224
 manpath: 222, 224
 mapscrn: 205, 206
 mcookie: 227, 230
 md5sum: 188, 190
 mesg: 227, 230
 meson: 187, 187
 mkdir: 188, 191
 mke2fs: 232, 234
 mkfifo: 188, 191
 mkfs: 227, 230
 mkfs.bfs: 227, 230
 mkfs.cramfs: 227, 230
 mkfs.ext2: 232, 234
 mkfs.ext3: 232, 234
 mkfs.ext4: 232, 234
 mkfs.minix: 227, 230
 mklost+found: 232, 234
 mknod: 188, 191
 mkswap: 227, 230
 mktemp: 188, 191
 mk_cmds: 232, 234
 mmroff: 197, 198
 modinfo: 177, 178
 modprobe: 177, 178

more: 227, 230
 mount: 227, 230
 mountpoint: 227, 230
 msgattrib: 155, 155
 msgcat: 155, 155
 msgcmp: 155, 155
 msgcomm: 155, 156
 msgconv: 155, 156
 msgen: 155, 156
 msgexec: 155, 156
 msgfilter: 155, 156
 msgfmt: 155, 156
 msggrep: 155, 156
 msginit: 155, 156
 msgmerge: 155, 156
 msgunfmt: 155, 156
 msguniq: 155, 156
 mtrace: 101, 107
 mv: 188, 191
 namei: 227, 230
 ncursesw6-config: 150, 151
 neqn: 197, 198
 newgidmap: 140, 143
 newgrp: 140, 143
 newuidmap: 140, 143
 newusers: 140, 143
 ngettext: 155, 156
 nice: 188, 191
 ninja: 186, 186
 nl: 188, 191
 nm: 128, 129
 nohup: 188, 191
 nologin: 140, 143
 nproc: 188, 191
 nroff: 197, 198
 nsenter: 227, 230
 nstat: 203, 204
 numfmt: 188, 191
 objcopy: 128, 129
 objdump: 128, 129
 od: 188, 191
 openssl: 175, 176
 openvt: 205, 206
 partx: 227, 230
 passwd: 140, 143
 paste: 188, 191
 patch: 210, 210
 pathchk: 188, 191
 pcprofiledump: 101, 107
 pdfmom: 197, 198
 pdffroff: 197, 198
 pdftexi2dvi: 212, 213
 peekfd: 154, 154
 perl: 168, 169
 perl5.40.0: 168, 169
 perlbug: 168, 169
 perldoc: 168, 169
 perlivp: 168, 169
 perlthanks: 168, 169
 pfbtops: 197, 198
 pgrep: 225, 225
 pic: 197, 198
 pic2graph: 197, 198
 piconv: 168, 169
 pidof: 225, 225
 ping: 165, 166
 ping6: 165, 166
 pinky: 188, 191
 pip3: 181
 pivot_root: 227, 230
 pkgconf: 127, 127
 pkill: 225, 225
 pl2pm: 168, 169
 pldd: 101, 107
 pmap: 225, 225
 pod2html: 168, 169
 pod2man: 168, 169
 pod2texi: 212, 213
 pod2text: 168, 169
 pod2usage: 168, 169
 podchecker: 168, 169
 podselect: 168, 169
 post-grohtml: 197, 198
 poweroff: 236, 236
 pr: 188, 191
 pre-grohtml: 197, 198
 preconv: 197, 198
 printenv: 188, 191
 printf: 188, 191
 prlimit: 227, 230
 prove: 168, 169
 prtstat: 154, 154
 ps: 225, 225
 psfaddtable: 205, 206
 psfgettable: 205, 206
 psfstrietable: 205, 206
 psfxtable: 205, 206
 pslog: 154, 154
 pstree: 154, 154
 pstree.x11: 154, 154

ptar: 168, 169
 ptardiff: 168, 169
 ptargrep: 168, 170
 ptx: 188, 191
 pwck: 140, 143
 pwconv: 140, 143
 pwd: 188, 191
 pwdx: 225, 225
 pwunconv: 140, 143
 pydoc3: 181
 python3: 181
 ranlib: 128, 129
 readelf: 128, 130
 readlink: 188, 191
 readprofile: 227, 230
 realpath: 188, 191
 reinicializar: 236, 236
 recode-sr-latin: 155, 156
 refer: 197, 199
 rename: 227, 230
 renice: 227, 230
 reset: 150, 151
 resize2fs: 232, 234
 resizepart: 227, 230
 rev: 227, 230
 rfcill: 227, 230
 rm: 188, 191
 rmdir: 188, 191
 rmmmod: 177, 178
 roff2dvi: 197, 199
 roff2html: 197, 199
 roff2pdf: 197, 199
 roff2ps: 197, 199
 roff2text: 197, 199
 roff2x: 197, 199
 routel: 203, 204
 rtacct: 203, 204
 rtcwake: 227, 230
 rtmon: 203, 204
 rtpr: 203, 204
 rtstat: 203, 204
 runcon: 188, 191
 runlevel: 236, 236
 runttest: 126, 126
 rview: 214, 216
 rvim: 214, 216
 script: 227, 230
 scriptlive: 227, 230
 scriptreplay: 227, 230
 sdiff: 194, 194
 sed: 153, 153
 seq: 188, 191
 setarch: 227, 230
 setcap: 137, 137
 setfacl: 136, 136
 setfattr: 135, 135
 setfont: 205, 206
 setkeycodes: 205, 206
 setleds: 205, 206
 setmetamode: 205, 206
 setsid: 227, 230
 setterm: 227, 230
 setvtrgb: 205, 206
 sfdisk: 227, 230
 sg: 140, 143
 sh: 159, 160
 sha1sum: 188, 191
 sha224sum: 188, 191
 sha256sum: 188, 191
 sha384sum: 188, 191
 sha512sum: 188, 191
 shasum: 168, 170
 showconsolefont: 205, 206
 showkey: 205, 206
 shred: 188, 191
 shuf: 188, 191
 shutdown: 236, 236
 size: 128, 130
 slabtop: 225, 225
 sleep: 188, 191
 sln: 101, 107
 soelim: 197, 199
 sort: 188, 191
 sotruss: 101, 107
 splain: 168, 170
 split: 188, 192
 sprof: 101, 107
 ss: 203, 204
 stat: 188, 192
 stdbuf: 188, 192
 strings: 128, 130
 strip: 128, 130
 stty: 188, 192
 su: 140, 143
 sulogin: 227, 231
 sum: 188, 192
 swapon: 227, 231
 swaponoff: 227, 231
 swapon: 227, 231
 switch_root: 227, 231

sync: 188, 192
 sysctl: 225, 225
 syslogd: 235, 235
 tabs: 150, 151
 tac: 188, 192
 tail: 188, 192
 talk: 165, 166
 tar: 211, 211
 taskset: 227, 231
 tbl: 197, 199
 tc: 203, 204
 tclsh: 122, 123
 tclsh8.6: 122, 123
 tee: 188, 192
 telinit: 236, 236
 telnet: 165, 166
 test: 188, 192
 texi2dvi: 212, 213
 texi2pdf: 212, 213
 texi2any: 212, 213
 texindex: 212, 213
 tfmtodit: 197, 199
 tftp: 165, 166
 tic: 150, 151
 timeout: 188, 192
 tload: 225, 226
 toe: 150, 151
 top: 225, 226
 touch: 188, 192
 tput: 150, 151
 tr: 188, 192
 traceroute: 165, 166
 troff: 197, 199
 true: 188, 192
 truncate: 188, 192
 tset: 150, 152
 tsort: 188, 192
 tty: 188, 192
 tune2fs: 232, 234
 tzselect: 101, 107
 uclampset: 227, 231
 udev-hwdb: 219, 221
 udevadm: 219, 221
 udevd: 219, 221
 ul: 227, 231
 umount: 227, 231
 uname: 188, 192
 uname26: 227, 231
 uncompress: 202, 202
 unexpand: 188, 192
 unicode_start: 205, 206
 unicode_stop: 205, 207
 uniq: 188, 192
 unlink: 188, 192
 unlz4: 114, 114
 unlzma: 112, 113
 unshare: 227, 231
 unxz: 112, 113
 updatedb: 196, 196
 uptime: 225, 226
 useradd: 140, 143
 userdel: 140, 143
 usermod: 140, 143
 users: 188, 192
 utmpdump: 227, 231
 uuuid: 227, 231
 uuidgen: 227, 231
 uuidparse: 227, 231
 vdir: 188, 192
 vi: 214, 216
 view: 214, 216
 vigr: 140, 143
 vim: 214, 216
 vimdiff: 214, 216
 vimtutor: 214, 216
 vipw: 140, 143
 vmstat: 225, 226
 w: 225, 226
 wall: 227, 231
 watch: 225, 226
 wc: 188, 192
 wdctl: 227, 231
 whatis: 222, 224
 wheel: 184
 whereis: 227, 231
 who: 188, 192
 whoami: 188, 192
 wipefs: 227, 231
 x86_64: 227, 231
 xargs: 196, 196
 xgettext: 155, 156
 xmlwf: 164, 164
 xsubpp: 168, 170
 xtrace: 101, 107
 xxd: 214, 216
 xz: 112, 113
 xzcat: 112, 113
 xzcmp: 112, 113
 xzdec: 112, 113
 xzdiff: 112, 113

xzegrep: 112, 113
 xzfgrep: 112, 113
 xzgrep: 112, 113
 xzless: 112, 113
 xzmore: 112, 113
 yacc: 157, 157
 yes: 188, 192
 zcat: 202, 202
 zcmp: 202, 202
 zdiff: 202, 202
 zdump: 101, 107
 zegrep: 202, 202
 zfgrep: 202, 202
 zforce: 202, 202
 zgrep: 202, 202
 zic: 101, 107
 zipdetails: 168, 170
 zless: 202, 202
 zmore: 202, 202
 znew: 202, 202
 zramctl: 227, 231
 zstd: 115, 115
 zstdgrep: 115, 115
 zstdless: 115, 115

Expat: 171, 171
 ld-2.40.so: 101, 107
 libacl: 136, 136
 libanl: 101, 107
 libasprintf: 155, 156
 libattr: 135, 135
 libbfd: 128, 130
 libblkid: 227, 231
 libBrokenLocale: 101, 107
 libbz2: 110, 111
 libc: 101, 107
 libcap: 137, 137
 libcheck: 193, 193
 libcom_err: 232, 234
 libcrypt: 138, 139
 libcrypto.so: 175, 176
 libctf: 128, 130
 libctf-nobfd: 128, 130
 libc_malloc_debug: 101, 107
 libdl: 101, 107
 libe2p: 232, 234
 libelf: 179, 179
 libexpat: 164, 164
 libexpect-5.45.4: 124, 125
 libext2fs: 232, 234

libfdisk: 227, 231
 libffi: 180
 libfl: 121, 121
 libformw: 150, 152
 libg: 101, 107
 libgcc: 144, 148
 libgcov: 144, 148
 libgdbm: 162, 162
 libgdbm_compat: 162, 162
 libgettextlib: 155, 156
 libgettextpo: 155, 156
 libgettextsrc: 155, 156
 libgmp: 131, 132
 libgmpxx: 131, 132
 libgomp: 144, 148
 libgprofng: 128, 130
 libhistory: 117, 117
 libhwasan: 144, 148
 libitm: 144, 148
 libkmod: 177
 liblsan: 144, 148
 libltdl: 161, 161
 liblto_plugin: 144, 148
 liblz4: 114, 114
 liblzma: 112, 113
 libm: 101, 107
 libmagic: 116, 116
 libman: 222, 224
 libmandb: 222, 224
 libmcheck: 101, 108
 libmemusage: 101, 108
 libmenuw: 150, 152
 libmount: 227, 231
 libmpc: 134, 134
 libmpfr: 133, 133
 libmvec: 101, 108
 libncurses++w: 150, 152
 libncursesw: 150, 152
 libnsl: 101, 108
 libnss_*: 101, 108
 libopcodes: 128, 130
 libpanelw: 150, 152
 libpcprofile: 101, 108
 libpipeline: 208
 libpkgconf: 127, 127
 libproc-2: 225, 226
 libpsx: 137, 137
 libpthread: 101, 108
 libquadmath: 144, 148
 libreadline: 117, 118

libresolv: 101, 108
 librt: 101, 108
 libsframe: 128, 130
 libsmartcols: 227, 231
 libss: 232, 234
 libssl.so: 175, 176
 libssp: 144, 148
 libstdbuf: 188, 192
 libstdc++: 144, 148
 libstdc++exp: 144, 148
 libstdc++fs: 144, 148
 libsubid: 140, 143
 libsupc++: 144, 148
 libtcl8.6.so: 122, 123
 libtclstub8.6.a: 122, 123
 libtextstyle: 155, 156
 libthread_db: 101, 108
 libtsan: 144, 149
 libubsan: 144, 149
 libudev: 219, 221
 libutil: 101, 108
 libuuid: 227, 231
 liby: 157, 157
 libz: 109, 109
 libzstd: 115, 115
 preloadable_libintl: 155, 156

/etc/hosts: 251
 configurando: 249
 rede de comunicação: 241, 241
 /etc/hosts: 251
 configurando: 249
 rede de comunicação: 241, 241
 /etc/hosts: 251
 configurando: 249
 rc: 241, 241
 reinicializar: 241, 242
 enviar sinais: 241, 242
 configura relógio: 241, 242
 configurando: 254
 configura relógio: 241, 242
 configurando: 254
 troca, arquivos e partições: 241, 242
 sysctl: 241, 242
 sysklogd: 241, 242
 configurando: 258
 sysklogd: 241, 242
 configurando: 258
 modelo: 241, 242
 udev: 241, 242
 udev_retry: 241, 242
 dwp: 128, 129

/boot/config-6.10.5: 267, 272
 /boot/System.map-6.10.5: 267, 272
 /dev/*: 78
 /etc/fstab: 265
 /etc/group: 81
 /etc/hosts: 251
 /etc/inittab: 253
 /etc/inputrc: 262
 /etc/ld.so.conf: 106
 /etc/lfs-release: 276
 /etc/localtime: 105
 /etc/lsb-release: 276
 /etc/mke2fs.conf: 233
 /etc/modprobe.d/usb.conf: 271
 /etc/nsswitch.conf: 105
 /etc/os-release: 276
 /etc/passwd: 81
 /etc/profile: 261
 /etc/protocols: 100
 /etc/resolv.conf: 250
 /etc/services: 100
 /etc/syslog.conf: 235
 /etc/udev: 219, 221
 /etc/udev/hwdb.bin: 220

checkfs: 241, 241
 cleanfs: 241, 241
 console: 241, 241
 configurando: 255
 console: 241, 241
 configurando: 255
 Criação de arquivo na inicialização
 configurando: 258
 funções: 241, 241
 parar: 241, 241
 hostname
 configurando: 251
 ifdown: 241, 241
 ifup: 241, 241
 ipv4-estático: 241, 242
 rede local: 241, 241
 /etc/hosts: 251
 rede local: 241, 241
 /etc/hosts: 251
 módulos: 241, 241
 montar sistemas de arquivos: 241, 241
 montar sistemas de arquivos virtuais: 241, 241
 rede de comunicação: 241, 241

/etc/vimrc: 215
/run/utmp: 81
/usr/include/asm-generic/*.h: 52, 52
/usr/include/asm/*.h: 52, 52
/usr/include/drm/*.h: 52, 52
/usr/include/linux/*.h: 52, 52
/usr/include/misc/*.h: 52, 52
/usr/include/mtd/*.h: 52, 52
/usr/include/rdma/*.h: 52, 52
/usr/include/scsi/*.h: 52, 52
/usr/include/sound/*.h: 52, 52
/usr/include/video/*.h: 52, 52
/usr/include/xen/*.h: 52, 52
/var/log/btmp: 81
/var/log/lastlog: 81
/var/log/wtmp: 81
/etc/shells: 263
páginas de manual: 99, 99