Linux From Scratch

Versão 11.1

Publicado 1º de março de 2022

Criado por Gerard Beekmans
Editor-chefe: Bruce Dubbs

Linux From Scratch: Versão 11.1: Publicado 1º de março de 2022

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

Copyright © 1999-2022 Gerard Beekmans

Direitos autorais © 1999-2022, Gerard Beekmans

Todos os direitos reservados.

Este livro é licenciado sob uma Creative Commons License.

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

Linux® é uma marca comercial registrada do Linus Torvalds.

Índice

Prefácio

Introdução

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

Há sempre várias maneiras de se completar uma tarefa. O mesmo pode ser dito sobre distribuições Linux. Muitas surgiram ao longo dos anos. Algumas ainda existem, outras se transformaram em outra distribuição, e ainda há outras que ficaram relegadas às nossas memórias. Todas elas executam as tarefas de maneira diferente para se adequar às necessidades de seus respectivos públicos-alvo. Devido ao fato de haver tantas maneiras de se executar uma tarefa, eu comecei a perceber que eu não tinha que me limitar à implementação de outra pessoa. Antes de descobrir o Linux, nós simplesmente lidávamos com problemas em outros Sistemas Operacionais como se não tivéssemos escolha. A coisa era o que era, não importando se você gostasse ou não. Com Linux, o conceito de escolha começou a emergir. Se você não gostou de alguma coisa, você seria livre, até encorajado, a mudá-la.

Eu tentei várias distribuições, mas não consegui me decidir por nenhuma. Elas eram ótimas distribuições em seu próprio direito. Não era mais uma questão de certo ou errado. O problema havia se transformado em uma questão de gosto pessoal. Com todas aquelas opções disponíveis, tornou-se aparente que não haveria um sistema que seria perfeito para mim. Então eu me propus a criar meu próprio sistema Linux que estaria totalmente em conformidade com minhas preferências pessoais.

Para realmente fazer meu próprio sistema, eu resolvi compilar tudo a partir do código fonte em vez de usar pacotes pré-compilados. Esse sistema Linux "perfeito" teria a força de vários sistemas sem suas fraquezas visíveis. A princípio, a ideia era bastante amedrontadora. Mas eu me mantive comprometido à ideia de que esse sistema poderia ser construído.

Após lidar com questões como dependências recíprocas e erros durante a compilação, eu finalmente construí um sistema Linux customizado. O sistema era totalmente operacional e perfeitamente utilizável como qualquer outro sistema Linux disponível na época. Mas era minha própria criação. Montar um sistema desses foi muito gratificante. A única coisa que poderia ser melhor seria se eu mesmo tivesse escrito cada programa. Essa foi a melhor coisa que se seguiu.

Conforme eu compartilhei meus objetivos e minhas experiências com outros membros da comunidade Linux, ficou aparente que havia um interesse firme nessas ideias. Logo ficou claro que tal sistema Linux customizado não serviria apenas para as necessidades específicas dos usuários, mas também como uma oportunidade ideal para programadores e administradores elevarem suas (existentes) habilidades com Linux. Como resultado desse interesse amplo, o Projeto *Linux From Scratch* nasceu.

Este livro Linux From Scratch é o núcleo do projeto. O livro provê a base e as instruções necessárias para você modelar e construir seu próprio sistema. Mesmo este livro disponibilizando instruções que resultarão 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 começar sua própria jornada.

Eu sinceramente espero que você se divirta trabalhando no seu próprio Linux From Scratch e aproveite os benefícios de ter um sistema 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 deste 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 prover é a habilidade de personalizar um sistema Linux para se ajustar às suas [de quem construir] próprias necessidades únicas.

Outro benefício chave de LFS é que ele te permite ter mais controle sobre o sistema sem confiar na implementação Linux de ninguém. Com LFS, você está no banco do motorista e dita cada aspecto do sistema.

LFS te permite criar sistemas muito compactos. Quando se instala distribuições regulares, você frequentemente é forçado a instalar muitos programas grandes os quais provavelmente nunca serão usados ou entendidos. Esses programas desperdiçam recursos. Você talvez argumente que, com os discos rígidos e CPUs de hoje, tais recursos não mais são uma consideração. As vezes, entretanto, você ainda está restrito por considerações de tamanho se nenhuma outra coisa. Pense acerca de CDs inicializáveis, mídias USB e sistemas embarcados. Essas são áreas onde LFS pode ser benéfico.

Outra vantagem de um sistema personalizado Linux construído é segurança. Ao compilar o sistema inteiro desde o zero, você está empoderado para auditar tudo e aplicar todas as correções de segurança desejadas. Não mais é necessário aguardar que outra pessoa compile os pacotes binários para consertar uma brecha de segurança. A menos que você examine a correção e a implemente você mesma(o), você não tem garantias de que o novo pacote binário foi construído corretamente e adequadamente conserta o problema.

A finalidade do [projeto] Linux From Scratch é a de construir um sistema em nível de fundação completo e utilizável. Se você não estiver afim de construir seu próprio sistema Linux desde o zero, então você talvez 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 poderosa das razões. Conforme você continue em sua experiência LFS, você descobrirá o poder que informação e conhecimento verdadeiramente trazem.

Arquiteturas Alvo do LFS

A principal arquitetura alvo do LFS são os processadores 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 os processadores Power PC e ARM. Para construir um sistema que utiliza uma dessas CPUs, o principal pré-requisito, em adição àqueles que estão nas próximas páginas, é uma distribuição Linux existente, como uma instalação LFS prévia, Ubuntu, Red Hat/Fedora, SuSE, ou outra distribuição que abranja a arquitetura que você tem. Note também que uma distribuição de 32-bits pode ser instalada e usada como um sistema hospedeiro em um computador AMD/Intel de 64-bits.

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

```
Arquitetura Tempo de Construção Tamanho de Construção
32-bit       239.9 minutos  3.6 GB
64-bit       233.2 minutos  4.4 GB
```

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

A construção de 64-bit padrão que é resultante do LFS é considerado um sistema 64-bit “puro”. Ou seja, ele suporta apenas executáveis 64-bit. Construir um sistema “multi-lib” [de múltiplas bibliotecas] exige a compilação de muitos aplicativos duas vezes, uma vez para um sistema de 32-bit e outra vez para um sistema de 64-bit. Isso não é diretamente suportado em LFS, pois interferiria no objetivo educacional de prover as instruções necessárias para um sistema Linux base estrito. Alguns editores de LFS/BLFS mantém uma bifurcação de LFS para multilib, que é 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. Essa tarefa exige um certo nível de conhecimento de administração de sistemas Unix para resolver problemas e corretamente executar os comandos listados. Em particular, no mínimo, você já deveria ter a habilidade de usar linha de comando (shell) para copiar ou mover arquivos e diretórios, listar diretórios e conteúdos de arquivos, e navegar entre os diretórios. Também é de se esperar que você tenha um conhecimento razoável sobre como usar e instalar software [em um sistema] Linux.

Devido ao fato do livro LFS assumir que você tem *pelo menos* esse nível básico de habilidades, os vários fóruns de suporte do LFS não serão adequados para ajudá-lo nessas áreas. Você vai perceber que suas perguntas com relação a esse conhecimento básico não serão respondidas ou serão remetidas à lista de itens essenciais de pré-leitura.

Antes de construir um sistema LFS nós recomendamos a leitura do seguinte:

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

  Esse é um guia compreensivo de como construir e instalar pacotes de software Unix “genéricos” no Linux. Embora tenha sido escrito há algum tempo, esse guia ainda fornece um bom resumo das técnicas básicas necessárias para construir e instalar programas.

- Beginner's Guide to Installing from Source *http://moi.vonos.net/linux/beginners-installing-from-source/*

  Esse guia fornece um bom sumário de habilidades básicas e de técnicas necessárias para construir software a partir do código fonte.

LFS e Padrões

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

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

  O LSB tem quatro padrões separados: Core, Desktop, Runtime Languages (linguagens em tempo de execução), e Imaging. Em adição às exigências genéricas, há as exigências específicas de cada arquitetura. Existem também duas áreas para uso experimental: Gtk3 e Graphics. LFS tenta ficar de acordo com as arquiteturas discutidas na sessão anterior.

Nota

Muitas pessoas não concordam com os requisitos do LSB. O principal propósito de definir tais requisitos é o de garantir que softwares proprietários possam ser instalados e executados adequadamente em um sistema que respeite o referido padrão. Sendo o LFS baseado em código fonte, a(o) usuária(o) tem total controle sobre quais pacotes quer e muitas(os) escolhem não instalar alguns dos pacotes especificados pelo LSB.

Criar um sistema LFS completo capaz de passar nos testes das certificações do LSB é possível, mas não sem muitos pacotes adicionais que estão além do escopo do LFS. Esses pacotes adicionais tem instruções para instalação no BLFS.

Pacotes disponibilizados pelo LFS que são necessários para satisfazer os requisitos do LSB

LSB Core:	Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, Grep, Gzip, M4, Man-DB, Ncurses, Procps, Psmisc, Sed, Shadow, Tar, Util-linux, Zlib
LSB Desktop:	Nenhum
LSB Runtime Languages:	Perl, Python
LSB Imaging:	Nenhum
LSB Gtk3 e Gráficos LSB (Uso Experimental):	Nenhum

Pacotes disponibilizados pelo BLFS necessários para satisfazer os requisitos do LSB

LSB Core:	At, Batch (uma parte de At), Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, PAM, Pax, Sendmail (ou Postfix ou Exim), time
LSB Desktop:	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, Xorg
LSB Runtime Languages:	Libxml2, Libxslt
LSB Imaging:	CUPS, Cups-filters, Ghostscript, SANE
LSB Gtk3 e Gráficos LSB (Uso Experimental):	GTK+3

Pacotes não suportados pelo LFS ou BLFS necessários para satisfazer os requisitos do LSB

LSB Core:	Nenhum
LSB Desktop:	Qt4 (mas Qt5 é fornecido)
LSB Runtime Languages:	Nenhum
LSB Imaging:	Nenhum
LSB Gtk3 e Gráficos LSB (Uso Experimental):	Nenhum

Justificativa para os pacotes no Livro

Como declarado anteriormente, a finalidade do [projeto] LFS é a de construir um sistema em nível de fundação completo e utilizável. Isso inclui todos os pacotes necessários para replicá-lo ao tempo que disponibiliza uma base relativamente pequena sobre a qual a(o) usuária(o) pode personalizar um sistema mais completo baseado nas escolhas da(o) usuária(o). Isso não significa que o LFS é o menor sistema possível. Vários pacotes importantes estão inclusos que não são estritamente necessários. As listas abaixo documentam a justificativa para cada pacote no livro.

- Acl

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

- Attr

  Esse pacote contém aplicativos para a administração de atributos estendidos sobre objetos do sistema de arquivos.

- Autoconf

  Esse pacote contém aplicativos para produzir shell scripts que podem configurar automaticamente o código fonte a partir de um modelo do desenvolvedor. É geralmente necessário para reconstruir um pacote após atualizações para os procedimentos de construção.

- Automake

  Esse pacote contém aplicativos para gerar arquivos Make a partir de um modelo. É geralmente necessário para reconstruir um pacote após atualizações para os procedimentos de construção.

- Bash

  Esse pacote satisfaz um requisito central do LSB para disponibilizar uma interface Bourne Shell para o sistema. Foi escolhido em vez de outros pacotes de shell pelo seu uso comum e extensas capacidades que transcendem as funções básicas do shell.

- Bc

  Esse pacote disponibiliza uma linguagem de processamento numérico com precisão arbitrária. Ele satisfaz requisitos necessários quando da construção do Kernel do Linux.

- Binutils

  Esse pacote contém um linker, um assembler e outras ferramentas para manipular arquivos objeto. Os aplicativos nesse pacote são necessários para compilar a maioria dos pacotes em um sistema LFS e além.

- Bison

  Esse pacote contém a versão GNU do yacc (Yet Another Compiler Compiler) necessário para construir vários outros aplicativos no LFS.

- Bzip2

  Esse pacote contém aplicativos para compressão e descompressão de arquivos. É necessário para descomprimir muitos pacotes do LFS.

- Check

  Esse pacote contém um conjunto de ferramentas de teste para outros aplicativos.

- Coreutils

  Esse pacote contém um número de aplicativos essenciais para visualização e manipulação de arquivos e de 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 em LFS.

- DejaGNU

  Esse pacote contém um sistema para testar outros aplicativos.

- Diffutils

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

- E2fsprogs

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

- Eudev

  Esse pacote é um gerenciador de dispositivo. Ele controla dinamicamente o dono, permissões, nomes, e links simbólicos de dispositivos no diretório /dev conforme dispositivos são adicionados ou removidos do sistema.

- Expat

  Esse pacote contém uma biblioteca relativamente pequena de análise de XML. Ela é exigida pelo módulo de Perl XML::Parser.

- Expect

  Esse pacote contém um aplicativo para execução de scripts de diálogos 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 em seus scripts de construção.

- Findutils

  Esse pacote contém 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 textos. É a versão GNU do aplicativo lex (lexical analyzer). É necessário para construir vários pacotes do LFS.

- Gawk

  Esse pacote contém 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 pacote é o Gnu Compiler Collection. Ele contém os compiladores C e C++ assim como vários outros não construídos por LFS.

- GDBM

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

- Gettext

  Esse pacote contém utilitários e bibliotecas para internacionalização e localização de numerosos pacotes.

- Glibc

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

- GMP

Esse pacote contém bibliotecas matemáticas que fornecem funções úteis para aritmética de precisão arbitrária. É necessário para compilar GCC.

- Gperf

  Esse pacote contém um aplicativo que gera uma função perfeita de hash a partir de uma chave configurada. Ele é exigido por Eudev.

- Grep

  Esse pacote contém aplicativos para procurar dentro de arquivos. Esses aplicativos são usados pela maioria dos scripts de construção dos pacotes.

- Groff

  Esse pacote contém aplicativos para processamento e formatação de texto. Uma função importante desses aplicativos é a de formatar páginas de manual.

- GRUB

  Esse pacote é o Grand Unified Boot Loader. Ele é um dos vários gerenciadores de inicialização disponíveis, mas é o mais flexível.

- Gzip

  Esse pacote contém aplicativos para compressão e descompressão de arquivos. Ele é necessário para descomprimir muitos pacotes em LFS.

- Iana-etc

  Esse pacote fornece dados para serviços e protocolos de rede. Ele é necessário para habilitar suporte a rede adequado.

- Inetutils

  Esse pacote contém aplicativos para administração básica de rede.

- Intltool

  Esse pacote contém ferramentas para a extração de sequências de caracteres traduzíveis a partir de arquivos fonte.

- IProute2

  Esse pacote contém aplicativos para redes IPv4 e IPv6 básicas e avançadas. Ele foi escolhido em vez de outros pacotes comuns de ferramentas de rede (net-tools) pelo seu suporte a IPv6.

- Kbd

  Esse pacote contém arquivos de tabelas chave, utilitários de teclados que não são estadunidenses, e um número de fontes de console.

- Kmod

  Esse pacote contém aplicativos necessários para administrar os módulos de kernel do Linux.

- Less

  Esse pacote contém um visualizador de textos muito bom que permite rolar para cima ou para baixo quando se visualiza um arquivo. Ele também é usado pelo Man-DB para visualizar páginas de manual.

- Libcap

Esse pacote implementa as interfaces do espaço de usuário para as capacidades POSIX 1003.1e disponíveis em kernels Linux.

- Libelf

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

- Libffi

  Esse pacote implementa uma interface de programação portável, de alto nível, para várias convenções de chamada. Alguns aplicativos talvez não saibam, ao tempo da compilação, quais argumentos são para serem passados para uma função. Por exemplo, um interpretador talvez 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 contém 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 portável. Ele é necessário para as ferramentas de testes em outros pacotes do LFS.

- Linux Kernel

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

- M4

  Esse pacote contém 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 em LFS.

- Man-DB

  Esse pacote contém aplicativos para encontrar e visualizar páginas de manual. Ele foi escolhido em vez do pacote man devido a capacidades superiores de internacionalização. Ele faz as vezes do aplicativo man.

- Man-pages

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

- Meson

  Esse pacote fornece uma ferramenta de software para automatizar a construção de software. A finalidade principal para Meson é a de minimizar a quantidade de tempo que desenvolvedores de software precisam investir configurando o sistema de construção deles. Ele é exigido para construir Systemd, bem como muitos pacotes BLFS.

- MPC

Esse pacote contém funções para a aritmética de números complexos. Ele é exigido por GCC.

- MPFR

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

- Ninja

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

- Ncurses

  Esse pacote contém bibliotecas para manipulação independente de terminal de telas de carácter. Ele é frequentemente usado para fornecer controle de cursor para um sistema com menus. Ele é necessitado por um número de pacotes em LFS.

- Openssl

  Esse pacote fornece ferramentas e bibliotecas de gerenciamento relacionadas a criptografia. Essas são úteis para fornecer funções criptográficas para outros pacotes, incluindo o kernel Linux.

- Patch

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

- Perl

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

- Pkg-config

  Esse pacote fornece um aplicativo que retorna metadados acerca de uma biblioteca ou pacote instalado.

- Procps-NG

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

- Psmisc

  Esse pacote contém 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 desenho que enfatiza a legibilidade de código.

- Readline

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

- Sed

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

- Shadow

  Esse pacote contém aplicativos para manipulação de senhas de uma maneira segura.

- Sysklogd

  Esse pacote contém aplicativos para registro de mensagens do sistema, tais como aqueles enviadas pelo kernel ou por processos deamons quando eventos não-usuais acontecem.

- Sysvinit

  Esse pacote fornece o aplicativo init, o qual é o pai de todos os outros processos no sistema Linux.

- Tar

  Esse pacote fornece capacidades de empacotamento e de extração de virtualmente todos os pacotes usados em LFS.

- Tcl

  Esse pacote contém a Tool Command Language usada em muitas ferramentas de teste em pacotes do LFS.

- Texinfo

  Esse pacote contém aplicativos para leitura, escrita e conversão de páginas info. Ele é usado nos procedimentos de instalação de muitos pacotes LFS.

- Util-linux

  Esse pacote contém uma variedade de aplicativos utilitários. Entre eles estão utilitários para manipulação de sistemas de arquivos, consoles, partições e mensagens.

- Vim

  Esse pacote contém um editor. Ele foi escolhido por causa da compatibilidade com o clássico editor vi e o seu número gigante de capacidades poderosas. Um editor é uma escolha muito pessoal para muitas(os) usuárias(os) e qualquer outro editor poderia ser substituído se assim desejar.

- XML::Parser

  Esse pacote é um módulo Perl que interage com Expat.

- XZ Utils

  Esse pacote contém aplicativos para compressão e descompressão de arquivos. Ele fornece a maior compressão geralmente disponível e é útil para descomprimir pacotes nos formatos XZ ou LZMA.

- Zlib

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

- Zstd

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

Tipografia

Para fazer as coisas mais fáceis de serem seguidas, existem algumas convenções tipográficas usadas neste livro. Esta sessão contém alguns exemplos da formatação tipográfica encontrada ao longo de Linux From Scratch.

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

Essa forma de texto é desenhada para ser digitada do jeito que está, a menos que seja dito o contrário no texto que a envolve. É também usada na sessão 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
```

Note que a barra invertida deve ser seguida imediatamente por uma quebra de linha. Outros espaços em branco como tabulação criarão resultados incorretos.

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

Essa forma de texto (largura fixa) mostra a saída em tela, geralmente como resultado de um comando executado. Esse formato é também utilizado para mostrar nomes de arquivos, como `/etc/ld.so.conf`.

Emphasis

Essa forma de texto é usada para vários propósitos neste livro. Seu propósito principal é o de enfatizar pontos ou itens importantes.

https://www.linuxfromscratch.org/

Esse formato é usado para hiperlinks tanto dentro da comunidade LFS e para páginas externas. Isso inclui HOWTOs, locais de downloads 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 para o sistema criar o arquivo `$LFS/etc/group` a partir do que seja digitado nas linhas seguintes até encontrar a sequência “End Of File” (EOF). Portanto, toda essa sequência é geralmente digitada da maneira como é vista.

<REPLACED TEXT>

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

[OPTIONAL TEXT]

Esse formato é usado para encapsular texto que é opcional.

`passwd(5)`

Esse formato é usado para referir-se a uma página de manual específica (man). O número entre parênteses indica uma seção específica dentro dos manuais. Por exemplo, **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** exibirá a primeira página de manual que corresponde a “passwd”, a qual será `/usr/share/man/man1/passwd.1`. Para esse exemplo, você precisará executar **man 5 passwd** para ler a página sendo especificada. Perceba que a maioria das páginas de manual não tem nomes duplicados de páginas em diferentes seções. Portanto, **man *<nome do aplicativo>*** geralmente é suficiente.

Estrutura

Este livro é dividido nas seguintes partes.

Parte I – Introdução

A Parte I explica algumas notas importantes sobre como proceder com a instalação do LFS. Essa seção também fornece metainformação sobre o 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 Cruzado de Ferramentas do LFS e Ferramentas Temporárias

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

Parte IV - Construindo o Sistema LFS

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

Parte V - Apêndices

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 de pacotes, aplicativos, bibliotecas, e scripts.

Errata e Avisos de Segurança

Os aplicativos utilizados para criar um sistema LFS estão sendo constantemente atualizados e melhorados. Alertas de segurança e correções de defeitos talvez se tornem disponíveis após o livro LFS ter sido lançado. Para checar se versões de pacotes ou instruções neste lançamento de LFS necessitam de quaisquer modificações para acomodar vulnerabilidades de segurança ou outras correções de defeitos, por favor visite *https://www.linuxfromscratch.org/lfs/errata/11.1/* antes de continuar com a sua construção. Você deveria tomar nota de quaisquer mudanças mostradas e aplicá-las às seções relevantes do livro conforme você progride com a construção do sistema LFS.

Adicionalmente, os editores de Linux From Scratch mantém uma lista de vulnerabilidades de segurança descobertas depois que um livro foi lançado. Para checar se existem quaisquer vulnerabilidades ativas de segurança, por favor visite *https://www.linuxfromscratch.org/lfs/advisories/* antes de proceder com sua construção. Você deveria tomar nota de quaisquer conselhos e executar os passos para corrigir quaisquer vulnerabilidades de segurança conforme você progride com a construção do sistema LFS.

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, OpenMandriva, Fedora, ou openSUSE). Esse sistema Linux existente (o anfitrião) será usado como ponto de partida para fornecer os aplicativos necessários, incluindo um compilador, um vinculador, e um interpretador de comandos, para construir o novo sistema. Selecione a opção “desenvolvimento” durante a instalação da distribuição para estar apto a acessar essas ferramentas.

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

Capítulo 2 deste livro descreve como criar uma nova partição Linux nativa e sistema de arquivos. Esse é o local onde o novo sistema LFS será compilado e instalado. Capítulo 3 explica quais pacotes e patches precisam ser baixados para construir um sistema LFS e como eles devem ser armazenados no novo sistema de arquivos. 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 sobre os quais você deve estar ciente antes de começar seu trabalho ao longo do Capítulo 5 e além.

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.

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

Capítulo 7 então entra em um ambiente "chroot" e usa as ferramentas previamente construídas para construir as ferramentas adicionais necessárias para construir e para testar o sistema final.

Esse esforço para isolar o sistema novo do sistema anfitrião talvez pareça excessivo. Uma explicação técnica completa sobre o porquê isso é feito é fornecida em Notas Técnicas do Conjunto de Ferramentas.

Em Capítulo 8, o sistema LFS completo é construído. Outra vantagem fornecida pelo ambiente chroot é que ele te permite continuar usando o sistema anfitrião enquanto que 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 em Capítulo 9, e o kernel e carregador de inicialização são configurados em Capítulo 10. Capítulo 11 contém informação sobre como continuar a experiência LFS além deste livro. Após os passos neste livro terem sido implementados, o computador estará pronto para reiniciar no novo sistema LFS.

Esse é o processo em poucas palavras. Informação detalhada sobre cada passo é discutida nos capítulos seguintes e nas descrições dos pacotes. Itens que talvez pareçam complicados 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 último lançamento

Nesta versão de LFS, houve uma grande reorganização do livro usando técnicas que evitam a modificação do sistema anfitrião e fornecem um seguimento mais estrito do processo de construção.

Abaixo está uma lista das atualizações de pacotes feitas desde o lançamento anterior do livro.

Atualizado para:

-
- Automake-1.16.5

- Bash-5.1.16
- Bc-5.2.2
- Binutils-2.38
- Bison-3.8.2
- Coreutils-9.0
- E2fsprogs-1.46.5
- Eudev-3.2.11
- Expat-2.4.6
- File-5.41
- Findutils-4.9.0
- Gawk-5.1.1
- GDBM-1.23
- Glibc-2.35
- Gzip-1.11
- IANA-Etc-20220207
- Inetutils-2.2
- IPRoute2-5.16.0
- Libcap-2.63
- Libelf-0.186 (de: elfutils)
- Libpipeline-1.5.5
- Linux-5.16.9
- Man-DB-2.10.1
- Meson-0.61.1
- Ncurses-6.3
- Openssl-3.0.1
- Python-3.10.2
- Readline-8.1.2
- Shadow-4.11.1
- SysVinit-3.01
- Tcl-8.6.12
- Tzdata-2021e
- Util-Linux-2.37.4
- Vim-8.2.4383
- Zstd-1.5.2

Adicionado:

-
- binutils-2.38-lto_fix-1.patch
- coreutils-9.0-chmod_fix-1.patch
- file-5.40-upstream_fixes-1.patch

- shadow-4.10-useradd_segfault-1.patch
- sysvinit-3.01-consolidated-1.patch

1.3. Registro de Mudanças

Esta é a versão 11.1 do livro Linux From Scratch, datada de 1º de março de 2022. Se este livro estiver com mais de seis meses, então uma versão nova e melhor provavelmente já está disponível. Para descobrir, por favor verifique um dos sites via *https://www.linuxfromscratch.org/mirrors.html*.

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

Entradas de Registro de Mudanças:

- 2022-03-01
  - [bdubbs] - LFS-11.1 lançado.
- 2022-02-23
  - [bdubbs] - Atualização para expat-2.4.6 (correção de segurança). Corrige *#5011*.
- 2022-02-15
  - [bdubbs] - LFS-11.1-rc1 lançado.
  - [bdubbs] - Adicionar binutils-2.38 LTO patch. Corrige *#5011*.
  - [bdubbs] - Atualização para util-linux-2.37.4. Corrige *#5010*.
  - [bdubbs] - Atualização para man-db-2.10.1. Corrige *#5009*.
  - [bdubbs] - Atualização para linux-5.16.9. Corrige *#5008*.
  - [bdubbs] - Atualização para vim-8.2.4383 (Atualização de Segurança). Endereça *#4500*.
  - [bdubbs] - Atualização para iana-etc-20220207. Endereça *#5006*.
- 2022-02-10
  - [xry111] - Contorna um problema que causa os binários se vincularem a bibliotecas da distribuição anfitriã para a passagem dois de binutils. Agora é desnecessário construir zlib em capítulo 6.
- 2022-02-09
  - [bdubbs] - Atualização para bc-5.2.2. Corrige *#5004*.
  - [bdubbs] - Atualização para linux-5.16.8. Corrige *#5005*.
  - [bdubbs] - Atualização para binutils-2.38. Exige a adição de zlib para Capítulo 6. Corrige *#5007*.
- 2022-02-04
  - [xry111] - Remove diretivas **bash** +h em chroot. Corrige *#4998*.
  - [xry111] - Atualização para man-db-2.10.0. Corrige *#5002*.
  - [xry111] - Move OpenSSL para antes de Kmod e habilita OpenSSL para construção de Kmod.
  - [xry111] - Atualização para gdbm-1.23. Corrige *#5000*.
  - [xry111] - Atualização para tcl-8.6.12. Corrige *#5001*.
  - [thomas] - Remove sed das instruções glibc em capítulo 8. Foi submetida ao desenvolvedor.
- 2022-02-03
  - [bdubbs] - Adicionado patch chmod do coreutils-9.0. Corrige *#4992*.

- [bdubbs] - Atualização para glibc-2.35. Corrige *#4999*.
- [bdubbs] - Atualização para linux-5.16.5. Corrige *#4996*.
- [bdubbs] - Atualização para findutils-4.9.0. Corrige *#4995*.
- [bdubbs] - Atualização para expat-2.4.4. Corrige *#4993*.
- [bdubbs] - Atualização para iana-etc-20220128. Corrige *#4994*.

- 2022-01-29
  - [bdubbs] - Atualização para linux-5.16.4. Corrige *#4991*.
- 2022-01-27
  - [bdubbs] - Atualização para vim-8.2.4236. Endereça *#4500*.
  - [bdubbs] - Atualização para zstd-1.5.2. Corrige *#4988*.
  - [bdubbs] - Atualização para util-linux-2.37.3 (correção de segurança). Corrige *#4989*.
  - [bdubbs] - Atualização para Python-3.10.2. Corrige *#4987*.
  - [bdubbs] - Atualização para linux-5.16.2. Corrige *#4979*.
  - [bdubbs] - Atualização para libcap-2.63. Corrige *#4990*.
  - [bdubbs] - Atualização para iproute2-5.16.0. Corrige *#4982*.
  - [bdubbs] - Atualização para iana-etc-20220120. Corrige *#4975*.
- 2022-01-20
  - [bdubbs] - Atualização para expat-2.4.3 (correções de segurança). Corrige *#4984*.
  - [pierre] - Atualização para meson-0.61.1. Corrige *#4985*.
- 2022-01-17
  - [thomas] - Adicionada uma correção de um erro de digitação para o patch de meson-0.61.0.
- 2022-01-15
  - [bdubbs] - Atualização para shadow-4.11.1. Corrige *#4976*.
  - [bdubbs] - Atualização para readline-8.1.2. Corrige *#4980*.
  - [bdubbs] - Atualização para meson-0.61.0. Corrige *#4983*.
  - [bdubbs] - Atualização para libpipeline-1.5.5. Corrige *#4977*.
  - [bdubbs] - Atualização para bash-5.1.16. Corrige *#4978*.
- 2022-01-01
  - [bdubbs] - Atualização para e2fsprogs-1.46.5. Corrige *#4974*.
  - [bdubbs] - Atualização para zstd-1.5.1. Corrige *#4972*.
  - [bdubbs] - Atualização para expat-2.4.2. Corrige *#4970*.
  - [bdubbs] - Atualização para shadow-4.10. Corrige *#4969*.
  - [bdubbs] - Atualização para sysvinit-3.01. Corrige *#4968*.
  - [bdubbs] - Atualização para linux-5.15.12. Corrige *#4967*.
  - [bdubbs] - Atualização para iana-etc-20211224. Corrige *#4962*.
  - [bdubbs] - Atualização para openssl-3.0.1. Corrige *#4922*.

  - [bdubbs] - Atualização para eudev-3.2.11. Corrige *#4914*.
- 2021-12-30
  - [renodr] - Atualização para meson-0.60.3. Corrige *#4973*.
- 2021-12-15
  - [bdubbs] - Atualização para python3-3.10.1. Corrige *#4963*.
  - [bdubbs] - Atualização para openssl-1.1.1m. Corrige *#4966*.
  - [bdubbs] - Atualização para linux-5.15.7. Corrige *#4964*.
  - [bdubbs] - Atualização para libcap-2.62. Corrige *#4965*.
- 2021-12-14
  - [thomas] - Permite a construção de findutils em sistemas de 32 bits. Commits aplicados a partir da branch multilib por [pierre].
- 2021-12-01
  - [bdubbs] - Atualização para vim-8.2.3704. Endereça *#4500*.
  - [bdubbs] - Atualização para iana-etc-20211124. Corrige *#4957*.
  - [bdubbs] - Atualização para bc-5.2.1. Corrige *#4959*.
  - [bdubbs] - Atualização para meson-0.60.2. Corrige *#4960*.
  - [bdubbs] - Atualização para linux-5.15.5. Corrige *#4956*.
- 2021-11-15
  - [bdubbs] - Atualização para iana-etc-20211112. Corrige *#4955*.
  - [bdubbs] - Atualização para elfutils-0.186. Corrige *#4954*.
  - [bdubbs] - Atualização para bc-5.2.0. Corrige *#4952*.
  - [bdubbs] - Atualização para ncurses-6.3. Corrige *#4951*.
  - [bdubbs] - Atualização para libpipeline-1.5.4. Corrige *#4950*.
  - [bdubbs] - Atualização para meson-0.60.1. Corrige *#4949*.
  - [bdubbs] - Atualização para iproute2-5.15.0. Corrige *#4948*.
  - [bdubbs] - Atualização para linux-5.15.2. Corrige *#4947*.
- 2021-11-01
  - [bdubbs] - Atualização para gawk-5.1.1. Corrige *#4946*.
  - [bdubbs] - Atualização para meson-0.60.0. Corrige *#4945*.
  - [bdubbs] - Atualização para libcap-2.60. Corrige *#4944*.
  - [bdubbs] - Atualização para gdbm-1.22. Corrige *#4943*.
  - [bdubbs] - Atualização para file-5.41. Corrige *#4942*.
  - [bdubbs] - Atualização para linux-5.14.15. Corrige *#4941*.
  - [bdubbs] - Atualização para iana-etc-20211025. Corrige *#4940*.
  - [bdubbs] - Atualização para tzdata-2021e. Corrige *#4939*.
- 2021-10-15

- [bdubbs] - Atualização para vim-8.2.3508. Endereça *#4500*.
- [bdubbs] - Atualização para tzdata-2021c. Corrige *#4934*.
- [bdubbs] - Atualização para Python-3.10.0. Corrige *#4938*.
- [bdubbs] - Atualização para Jinja2-3.0.2. Corrige *#4937*.
- [bdubbs] - Atualização para linux-5.14.12. Corrige *#4932*.
- [bdubbs] - Atualização para iana-etc-20211004. Corrige *#4933*.
- [bdubbs] - Atualização para bc-5.1.1. Corrige *#4936*.
- [bdubbs] - Atualização para automake-1.16.5. Corrige *#4935*.

- 2021-10-01
  - [bdubbs] - Atualização para vim-8.2.3458. Endereça *#4500*.
  - [bdubbs] - Atualização para iana-etc-20210924. Endereça *#4722*.
  - [bdubbs] - Atualização para tzdata-2021b. Corrige *#4929*.
  - [bdubbs] - Atualização para sysvinit-3.0.0. Corrige *#4927*.
  - [bdubbs] - Atualização para meson-0.59.2. Corrige *#4931*.
  - [bdubbs] - Atualização para linux-5.14.8. Corrige *#4925*.
  - [bdubbs] - Atualização para libcap-2.59. Corrige *#4926*.
  - [bdubbs] - Atualização para coreutils-9.0. Corrige *#4928*.
  - [bdubbs] - Atualização para bison-3.8.2. Corrige *#4930*.
- 2021-09-15
  - [bdubbs] - Garante que as instruções de documentação de tcl estão presentes. Corrige *#4923*.
  - [bdubbs] - Atualização para Python3-3.9.7. Corrige *#4916*.
  - [bdubbs] - Atualização para linux-5.14.3. Corrige *#4913*.
  - [bdubbs] - Atualização para libcap-2.57. Corrige *#4912*.
  - [bdubbs] - Atualização para iproute2-5.14.0. Corrige *#4917*.
  - [bdubbs] - Atualização para inetutils-2.2. Corrige *#4918*.
  - [bdubbs] - Atualização para gzip-1.11. Corrige *#4920*.
  - [bdubbs] - Atualização para gdbm-1.21. Corrige *#4919*.
  - [bdubbs] - Atualização para bison-3.8.1. Corrige *#4921*.
  - [bdubbs] - Atualização para bc-5.0.2. Corrige *#4905*.
- 2021-09-08
  - [renodr] - Corrige regressões em File que resultam em detecção inapropriada de texto e arquivos XZ.
- 2021-09-06
  - [bdubbs] - Esclarecimentos de texto na seção cópia de segurança/restauração de Capítulo 7. Grato a Kevin Buckley pelo patch.
- 2021-09-01
  - [bdubbs] - LFS-11.0 lançado.

1.4. Recursos

1.4.1. Perguntas Frequentes

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

1.4.2. Listas de Correio Eletrônico

O servidor `linuxfromscratch.org` 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 o FAQ não resolver o problema que você está tendo, 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 no 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 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 fazer com que o acesso ao site do projeto e o download dos pacotes exigidos seja mais conveniente. Por favor visite o site 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 questões e comentários para uma das listas de discussão (veja acima).

1.5. Ajuda

Se um problema ou uma pergunta for encontrado durante o trabalho com este 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 puder 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 podem ser facilmente respondidas indo para as Perguntas Frequentes e procurando nas listas de discussão primeiro. Então, para que nós possamos oferecer a melhor assistência possível, você precisa fazer alguma pesquisa por conta própria primeiro. 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 de ajuda.

1.5.1. Coisas a Mencionar

Além de uma breve explanação do problema sendo vivenciado, as coisas essenciais a incluir em qualquer pedido de ajuda são:

- A versão do livro sendo usado (neste caso 11.1)
- A distribuição anfitriã e versão sendo usada para criar LFS
- A saída do script Exigências do Sistema Anfitrião
- O pacote ou seção onde o problema foi encontrado
- A mensagem de erro exata ou o sintoma sendo recebido
- Nota se você se desviou do livro afinal

Nota

Desviar-se deste livro *não* significa que nós não vamos te ajudar. Afinal de contas, LFS é acerca de preferência pessoal. Ser sincero sobre quaisquer mudanças nos procedimentos estabelecidos nos ajuda a avaliar e determinar possíveis causas do seu problema.

1.5.2. Problemas de Script de Configuração

Se algo der errado quando executar o script **configure**, então revise o arquivo `config.log`. Esse arquivo talvez contenha erros encontrados durante [a execução de] **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 da tela quando o conteúdo de vários arquivos são úteis para determinar a causa de problemas de compilação. A saída da tela do script **configure** e do **make** executado podem ser úteis. Não é necessário incluir toda a saída, mas inclua informações relevantes suficientes. Abaixo está um exemplo do tipo de informação a incluir a partir da saída de tela do **make**:

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

Nesse caso, muitas pessoas incluiriam apenas a seção final:

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

Essa não é informação suficiente para diagnosticar adequadamente o problema, pois essa linha apenas mostra que algo deu errado, não *o quê* deu errado. A seção inteira, como no exemplo acima, é o que deveria ser salva, porque ela inclui o comando que foi executado e a(s) mensagem(ns) de erro associada(s).

Um artigo excelente sobre como pedir ajuda na Internet está disponível em *http://catb.org/~esr/faqs/smart-questions.html*. Leia e siga as dicas nesse documento para aumentar a possibilidade receber 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 construção de 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

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

Versões anteriores dos pacotes de software listados talvez 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 2.38 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-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk deveria ser um link para gawk)
- **GCC-4.8** incluindo o compilador C++, **g++** (Versões maiores que 11.2.0 não são recomendadas dado que elas não foram testadas). As bibliotecas C e C++ padrão (com cabeçalhos) também devem estar presentes, de forma que o compilador C++ possa construir aplicativos hospedados
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-3.2**

  A razão para a exigência da versão de kernel é que nós especificamos essa versão quando da construção de glibc em Capítulo 5 e Capítulo 8, por recomendação dos desenvolvedores. Ela também é exigida por udev.

  Se o kernel do anfitrião for anterior a 3.2, então você precisará substituir o kernel com uma versão mais atualizada. Existem duas maneiras de você fazer isso. Primeira, veja se seu fornecedor Linux fornece um pacote de kernel 3.2 ou mais atual. Se sim, então você talvez deseje instalá-lo. Se seu fornecedor não oferecer um pacote de kernel aceitável, ou você preferisse não instalá-lo, então você mesmo pode compilar um kernel. Instruções para a compilação de kernel e configuração de carregador de inicialização (presumindo que o anfitrião usa GRUB) estão localizadas em Capítulo 10.
- **M4-1.4.10**
- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**

- **Texinfo-4.7**
- **Xz-5.0.0**

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.) talvez funcionem, porém não são testados ou suportados pela equipe de desenvolvimento de LFS, e talvez exijam ou desvio das instruções ou correções adicionais para alguns pacotes.

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

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Script simples para listar números de versão de ferramentas críticas de desenvol
export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4
MYSH=$(readlink -f /bin/sh)
echo "/bin/sh -> $MYSH"
echo $MYSH | grep -q bash || echo "ERRO: /bin/sh não aponta para bash"
unset MYSH

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

if [ -h /usr/bin/yacc ]; then
  echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
elif [ -x /usr/bin/yacc ]; then
  echo yacc is `/usr/bin/yacc --version | head -n1`
else
  echo "yacc não encontrado"
fi

echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1

if [ -h /usr/bin/awk ]; then
  echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
elif [ -x /usr/bin/awk ]; then
  echo awk é `/usr/bin/awk --version | head -n1`
else
  echo "awk não encontrado"
fi
```

```
gcc --version | head -n1
g++ --version | head -n1
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
python3 --version
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1  # versão texinfo
xz --version | head -n1

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

bash version-check.sh
```

2.3. Construindo LFS em Estágios

LFS está desenhado 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 que estar pronto de uma só vez. O problema é que certos procedimentos tem que ser realizados outra vez após uma inicialização se retomando LFS em pontos diferentes.

2.3.1. Capítulos 1–4

Esses capítulos são realizados no sistema anfitrião. Quando da reinicialização, seja cuidadosa(o) com o seguinte:

- Procedimentos feitos como a(o) usuária(o) `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 deve estar montada.
- Esses dois capítulos *devem* ser feitos como a(o) usuária(o) `lfs`. Um **su - lfs** precisa ser feito antes de qualquer tarefa nesses capítulos. Ao falhar em fazer isso, 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 acerca da instalação de um pacote, então certifique-se de que qualquer arquivo tar descomprimido previamente foi removido, então extraia novamente os arquivos do pacote, e complete todas as instruções nessa seção.

2.3.3. Capítulos 7–10

- A partição /mnt/lfs deve estar montada.

- Umas poucas operações, de “Mudando Dono” até “Entrando no Ambiente Chroot” devem ser feitas como a(o) usuária(o) `root`, com a variável de ambiente LFS configurada para a(o) usuária(o) `root`.
- Quando entrar em chroot, a variável de ambiente LFS deve estar configurada para `root`. A variável LFS não mais é usada posteriormente.
- Os sistemas virtuais de arquivo devem estar montados. Isso pode ser feito antes ou depois de entrar em chroot mudando para um terminal virtual do anfitrião e, como `root`, executando os comandos em Seção 7.3.2, “Montando e Povoando /dev” e Seção 7.3.3, “Montando Sistemas de Arquivos Virtuais de Kernel”.

2.4. Criando uma Nova Partição

Como a maior parte dos outros sistemas operacionais, 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 arquivos tar dos códigos 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 espaço livre suficiente de armazenamento temporário. Adicionalmente, a compilação de pacotes pode exigir muito 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 **cfdisk** ou **fdisk** com uma opção de linha de comando indicando o disco rígido no qual a nova partição será criada—por exemplo `/dev/sda` para o controlador primário de disco. Crie uma partição nativa Linux e uma partição `swap`, se necessária. Por favor, recorra a `cfdisk(8)` ou `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 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) de LFS pela primeira vez.

Lembre-se da designação da nova partição (por exemplo, `sda5`). Este livro se referirá a essa como a partição 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

Pedidos de ajuda com relação a particionamento de disco frequentemente são enviados à lista de discussão do LFS. Esse é um assunto altamente subjetivo. O padrão para a maioria das distribuições é o de usar todo o disco com a exceção de uma pequena partição swap. Isso não é ideal para LFS por várias razões. Isso reduz flexibilidade; torna o compartilhamento de dados entre múltiplas distribuições ou construções LFS mais difícil; torna as cópias de segurança mais demoradas; e podem desperdiçar espaço de disco devido à alocação ineficiente de estruturas de sistema de arquivo.

2.4.1.1. A Partição Raiz

Uma partição raiz de 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 LFS e a maior parte de 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 espaço de disco for limitado, então mantenha a partição swap com dois (2) gigabytes e monitore a quantidade de troca de disco.

Se você quer usar a característica de 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 discos rígidos mecânicos você geralmente pode dizer se um sistema está usando swap simplesmente monitorando a atividade de disco e observando como o sistema reage a comandos. Para um drive SSD você não estará apta(o) a monitorar swap, porém você pode dizer quanto espaço de swap está sendo usado via aplicativos **top** ou **free**. O uso de um drive 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, deve ser criada se ela já não existir. Essa partição não é formatada, porém deve estar disponível para GRUB usar durante a instalação do carregador de inicialização. Essa partição normalmente será rotulada 'BIOS Boot' se usar **fdisk** ou terá um código de *EF02* se usar **gdisk**.

Nota

A Partição de Bios Grub deve estar no drive que o BIOS usa para inicializar o sistema. Esse não é necessariamente o mesmo drive onde a partição raiz de LFS está localizada. Discos em um sistema talvez usem tipos diferentes de tabela de partição. A exigência para essa partição 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 kernels e outras informações de inicialização. Para minimizar potenciais problemas de inicialização com discos maiores, torne essa a primeira partição física no seu primeiro controlador de disco. Um tamanho de partição de duzentos (200) megabytes é bastante adequado.
- /home – Altamente recomendada. Compartilhe seu diretório home e personalizações de usuário entre múltiplas distribuições ou construções LFS. O tamanho geralmente é bastante grande e depende do espaço de disco disponível.
- /usr – Em 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 LFS, uma partição separada para `/usr` normalmente não é necessária. Se você precisar dela de qualquer maneira, 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 tomar cuidado que um initramfs (não coberto por LFS) será necessário para inicializar um sistema com partição `/usr` separada.

- /opt – Esse diretório é mais útil para BLFS onde múltiplas instalações de pacotes grandes como Gnome ou KDE podem ser instalados sem embutir os arquivos na hierarquia /usr. Se usado, 5 a 10 gigabytes geralmente é adequado.
- /tmp – Um diretório /tmp separado é raro, mas útil ao se configurar um "thin client". Essa partição, se usada, geralmente não precisará exceder alguns gigabytes.
- /usr/src – Essa partição é muito útil para disponibilizar uma localização para armazenar os arquivos fontes de BLFS e compartilhá-los entre construções LFS. Ela também pode ser usada como uma localização para construir pacotes BLFS. Uma partição razoavelmente grande de 30 a 50 gigabytes permite muito espaço.

Qualquer partição separada que você queira que seja montada automaticamente durante a inicialização precisa ser especificada no `/etc/fstab`. Detalhes sobre como especificar partições serão discutidos em Seção 10.2, “Criando o Arquivo /etc/fstab”.

2.5. Criando um Sistema de Arquivos na Partição

Agora que uma partição em branco foi configurada, o sistema de arquivos pode ser criado. LFS pode usar qualquer sistema de arquivos reconhecido pelo kernel Linux, mas os tipos mais comuns são ext3 e ext4. A escolha do sistema de arquivos pode ser complexa e depende das características dos arquivos e o 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 journal para ajudar a recuperar o status da partição no caso de desligamento inadequado. É comumente usada como sistema de arquivos de propósito geral.

ext4
: é a versão mais nova da família de sistema de arquivos ext de tipos de partição. Ela fornece várias capacidades novas incluindo marcas temporais em nano segundos, criação e uso de arquivos muito grandes (16 TB), e melhoramentos de velocidade.

Outros sistemas de arquivos, incluindo FAT32, NTFS, ReiserFS, JFS, e XFS são úteis para propósitos especializados. Mais informação sobre esses sistemas de arquivos pode ser encontrada em *http://en.wikipedia.org/wiki/Comparison_of_file_systems*.

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

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

Substitua *<xxx>* com o nome da partição LFS.

Se você está 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*> com o 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 de 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 a escolha do diretório cabe totalmente a você. Se você está construindo LFS em uma partição separada, então esse diretório será o ponto de montagem para a partição. Escolha uma localização 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 $LFS/tools** os quais podem ser digitados literalmente. O interpretador de comandos automaticamente substituirá "$LFS" com "/mnt/lfs" (ou para o que a variável foi configurada) quando ele processar a linha de comando.

Cuidado

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

```
echo $LFS
```

Tenha certeza de que a saída mostra o caminho para sua localização de construção do sistema LFS, a qual é `/mnt/lfs` se o exemplo fornecido foi seguido. Se a saída 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 está configurada é editar o arquivo `.bash_profile` tanto em seu diretório home pessoal quanto em `/root/.bash_profile` e inserir o comando export acima. Adicionalmente, o interpretador de comandos especificado no arquivo `/etc/passwd` para todas(os) as(os) usuárias(os) que precisam da variável `LFS` precisa ser bash para assegurar que o arquivo `/root/.bash_profile` é 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` da(o) usuária(o) normalmente não é usado quando um terminal virtual é iniciado. Nesse caso, adicione o comando export ao arquivo `.bashrc` para a(o) usuária(o) e `root`. Adicionalmente, algumas distribuições tem instruções para não executar as instruções de `.bashrc` em uma invocação não interativa de bash. Certifique-se de adicionar o comando export antes do teste para uso não interativo.

2.7. Montando a Nova Partição

Agora que um sistema de arquivos foi criado, a partição precisa se tornar acessível. Para fazer isso, a partição precisa ser montada em um ponto de montagem escolhido. Para os propósitos deste livro, assume-se que o sistema de arquivos está montado sob o diretório especificado pela variável de ambiente `LFS` conforme descrito na seção anterior.

Crie o ponto de montagem e monte o sistema de arquivos de LFS executando:

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

Substitua *<xxx>* com a designação da partição LFS.

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

```
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>* com os nomes apropriados das partições.

Assegure-se de que essa nova partição não está montada com permissões que são muito restritivas (tais como as opções `nosuid` ou `nodev`). Execute o comando **mount** sem quaisquer parâmetros para ver quais opções estão configuradas para a partição LFS montada. Se `nosuid` e (ou) `nodev` estiverem configuradas, então a partição precisará ser remontada.

Atenção

As instruções acima assumem que você não estará reiniciando seu computador no decorrer do processo LFS. Se você desligar seu sistema, então você precisará remontar a partição LFS a cada vez que você reiniciar o processo de construção ou modificar seu arquivo /etc/fstab do sistema anfitrião para remontá-la automaticamente após inicialização. Por exemplo:

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

Se você usa partições adicionais opcionais, então certifique-se de adicioná-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>* com o nome da partição `swap`.

Agora que existe um lugar estabelecido para trabalhar, é tempo de baixar os pacotes.

Capítulo 3. Pacotes e Patches

3.1. Introdução

Este capítulo inclui uma lista de pacotes que precisam ser baixados para 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 talvez não funcionem com uma versão diferente, a menos que a versão diferente seja especificada por uma errata de LFS ou conselho de segurança. As versões mais novas de pacote talvez também tenham problemas que exigem contornos. Essas correções serão desenvolvidas e estabilizadas na versão de desenvolvimento do livro.

Para alguns pacotes, o tarball de lançamento e o tarball instantâneo de repositório (Git ou SVN) para este lançamento talvez seja publicado com nome semelhante de arquivo. Um tarball de lançamento contém arquivos generalizados (por exemplo, o script **configure** gerado por **autoconf**), em adição aos conteúdos do correspondente instantâneo de repositório. O livro usa tarballs de lançamento quando possível. Usar um instantâneo de repositório em vez de um tarball de lançamento especificado pelo livro causará problemas.

Localizações de downloads nem sempre podem estar acessíveis. Se uma localização de download mudou desde que 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 download em *https://www.linuxfromscratch.org/lfs/mirrors.html#files*.

Pacotes e patches 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 patches quanto como diretório de trabalho. Usando esse diretório, os elementos exigidos estarão localizados na partição 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 download:

```
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 patches necessários para construir LFS:

- Os arquivos podem ser baixados individualmente conforme descrito nas próximas duas seções.
- Para versões estáveis do livro, um tarball de todos os arquivos necessários pode ser baixado a partir de um dos espelhos de arquivos de LFS listados em *https://www.linuxfromscratch.org/mirrors.html#files*.
- Os arquivos podem ser baixados usando **wget** e uma lista wget conforme descrito abaixo.

Para baixar todos os pacotes e patches usando *lista-wget* como uma entrada para o comando **wget**, use:

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

Nota

O arquivo `lista-wget` mencionado acima recupera todos os pacotes para as versões sysV e systemd de LFS. Existe um total de cinco pacotes pequenos adicionais não necessários para o livro atual. O arquivo `md5sums` mencionado abaixo é específico para o livro atual.

Adicionalmente, começando com 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 esse 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.

3.2. Todos os Pacotes

Baixe ou de outra forma obtenha os seguintes pacotes:

• Acl (2.3.1) - 348 KB:
Home page: *https://savannah.nongnu.org/projects/acl*
Download: *https://download.savannah.gnu.org/releases/acl/acl-2.3.1.tar.xz*
MD5 sum: `95ce715fe09acca7c12d3306d0f076b2`

• Attr (2.5.1) - 456 KB:
Home page: *https://savannah.nongnu.org/projects/attr*
Download: *https://download.savannah.gnu.org/releases/attr/attr-2.5.1.tar.gz*
MD5 sum: `ac1c5a7a084f0f83b8cace34211f64d8`

• Autoconf (2.71) - 1,263 KB:
Home page: *https://www.gnu.org/software/autoconf/*
Download: *https://ftp.gnu.org/gnu/autoconf/autoconf-2.71.tar.xz*
MD5 sum: `12cfa1687ffa2606337efe1a64416106`

• Automake (1.16.5) - 1,565 KB:
Home page: *https://www.gnu.org/software/automake/*
Download: *https://ftp.gnu.org/gnu/automake/automake-1.16.5.tar.xz*
MD5 sum: `4017e96f89fca45ca946f1c5db6be714`
SHA256 sum: `80facc09885a57e6d49d06972c0ae1089c5fa8f4d4c7cfe5baea58e5085f136d`

• Bash (5.1.16) - 10,277 KB:
Home page: *https://www.gnu.org/software/bash/*
Download: *https://ftp.gnu.org/gnu/bash/bash-5.1.16.tar.gz*
MD5 sum: `c17b20a09fc38d67fb303aeb6c130b4e`

• Bc (5.2.2) - 428 KB:
Home page: *https://git.yzena.com/gavin/bc*
Download: *https://github.com/gavinhoward/bc/releases/download/5.2.2/bc-5.2.2.tar.xz*
MD5 sum: `632344cdb052af0e06087bd3b0034126`

• Binutils (2.38) - 23,098 KB:
Home page: *https://www.gnu.org/software/binutils/*
Download: *https://ftp.gnu.org/gnu/binutils/binutils-2.38.tar.xz*
MD5 sum: `6e39cad1bb414add02b5b1169c18fdc5`

• Bison (3.8.2) - 2,752 KB:
Home page: *https://www.gnu.org/software/bison/*
Download: *https://ftp.gnu.org/gnu/bison/bison-3.8.2.tar.xz*
MD5 sum: `c28f119f405a2304ff0a7ccdcc629713`

• Bzip2 (1.0.8) - 792 KB:
Download: *https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz*
MD5 sum: `67e051268d0c475ea773822f7500d0e5`

• Check (0.15.2) - 760 KB:
Home page: *https://libcheck.github.io/check*
Download: *https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz*
MD5 sum: `50fcafcecde5a380415b12e9c574e0b2`

• Coreutils (9.0) - 5,482 KB:
Home page: *https://www.gnu.org/software/coreutils/*
Download: *https://ftp.gnu.org/gnu/coreutils/coreutils-9.0.tar.xz*
MD5 sum: `0d79ae8a6124546e3b94171375e5e5d0`

• DejaGNU (1.6.3) - 608 KB:
Home page: *https://www.gnu.org/software/dejagnu/*
Download: *https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.3.tar.gz*
MD5 sum: `68c5208c58236eba447d7d6d1326b821`

• Diffutils (3.8) - 1,548 KB:
Home page: *https://www.gnu.org/software/diffutils/*
Download: *https://ftp.gnu.org/gnu/diffutils/diffutils-3.8.tar.xz*
MD5 sum: `6a6b0fdc72acfe3f2829aab477876fbc`

• E2fsprogs (1.46.5) - 9,307 KB:
Home page: *http://e2fsprogs.sourceforge.net/*
Download: *https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.46.5/e2fsprogs-1.46.5.tar.gz*
MD5 sum: `3da91854c960ad8a819b48b2a404eb43`

• Elfutils (0.186) - 9,015 KB:
Home page: *https://sourceware.org/elfutils/*
Download: *https://sourceware.org/ftp/elfutils/0.186/elfutils-0.186.tar.bz2*
MD5 sum: `2c095e31e35d6be7b3718477b6d52702`

• Eudev (3.2.11) - 2,075 KB:
Download: *https://github.com/eudev-project/eudev/releases/download/v3.2.11/eudev-3.2.11.tar.gz*
MD5 sum: `417ba948335736d4d81874fba47a30f7`

• Expat (2.4.6) - 444 KB:
Home page: *https://libexpat.github.io/*
Download: *https://prdownloads.sourceforge.net/expat/expat-2.4.6.tar.xz*
MD5 sum: `22a30c888752fdda9f8dd1b7281c54b0`

Nota

A(O) Desenvolvedora(or) talvez remova tarballs dos lançamentos específicos de Expat quando esses lançamentos contenham uma vulnerabilidade de segurança. Você deveria se referir a *Avisos de Segurança de LFS* para saber qual versão (com a vulnerabilidade corrigida) deveria ser usada. Você talvez baixe a versão vulnerável a partir de um espelho, porém isso não é recomendado.

• Expect (5.45.4) - 618 KB:
Home page: *https://core.tcl.tk/expect/*
Download: *https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz*
MD5 sum: `00fce8de158422f5ccd2666512329bd2`

• File (5.41) - 1040 KB:
Home page: *https://www.darwinsys.com/file/*
Download: *https://astron.com/pub/file/file-5.41.tar.gz*
MD5 sum: `18233bb0a0089dfdc7dfbc93b96f231b`

• Findutils (4.9.0) - 1,999 KB:
Home page: *https://www.gnu.org/software/findutils/*
Download: *https://ftp.gnu.org/gnu/findutils/findutils-4.9.0.tar.xz*
MD5 sum: `4a4a547e888a944b2f3af31d789a1137`

• Flex (2.6.4) - 1,386 KB:
Home page: *https://github.com/westes/flex*
Download: *https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz*
MD5 sum: `2882e3179748cc9f9c23ec593d6adc8d`

• Gawk (5.1.1) - 3,075 KB:
Home page: *https://www.gnu.org/software/gawk/*
Download: *https://ftp.gnu.org/gnu/gawk/gawk-5.1.1.tar.xz*
MD5 sum: `83650aa943ff2fd519b2abedf8506ace`

• GCC (11.2.0) - 78,996 KB:
Home page: *https://gcc.gnu.org/*
Download: *https://ftp.gnu.org/gnu/gcc/gcc-11.2.0/gcc-11.2.0.tar.xz*
MD5 sum: `31c86f2ced76acac66992eeedce2fce2`
SHA256 sum: `d08edc536b54c372a1010ff6619dd274c0f1603aa49212ba20f7aa2cda36fa8b`

• GDBM (1.23) - 1,092 KB:
Home page: *https://www.gnu.org/software/gdbm/*
Download: *https://ftp.gnu.org/gnu/gdbm/gdbm-1.23.tar.gz*
MD5 sum: `8551961e36bf8c70b7500d255d3658ec`

• Gettext (0.21) - 9,487 KB:
Home page: *https://www.gnu.org/software/gettext/*
Download: *https://ftp.gnu.org/gnu/gettext/gettext-0.21.tar.xz*
MD5 sum: `40996bbaf7d1356d3c22e33a8b255b31`

• Glibc (2.35) - 17,741 KB:
Home page: *https://www.gnu.org/software/libc/*
Download: *https://ftp.gnu.org/gnu/glibc/glibc-2.35.tar.xz*
MD5 sum: `dd571c67d85d89d7f60b854a4e207423`

• GMP (6.2.1) - 1,980 KB:
Home page: *https://www.gnu.org/software/gmp/*
Download: *https://ftp.gnu.org/gnu/gmp/gmp-6.2.1.tar.xz*
MD5 sum: `0b82665c4a92fd2ade7440c13fcaa42b`

• Gperf (3.1) - 1,188 KB:
Home page: *https://www.gnu.org/software/gperf/*
Download: *https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz*
MD5 sum: `9e251c0a618ad0824b51117d5d9db87e`

• Grep (3.7) - 1,603 KB:
Home page: *https://www.gnu.org/software/grep/*
Download: *https://ftp.gnu.org/gnu/grep/grep-3.7.tar.xz*
MD5 sum: `7c9cca97fa18670a21e72638c3e1dabf`

• Groff (1.22.4) - 4,044 KB:
Home page: *https://www.gnu.org/software/groff/*
Download: *https://ftp.gnu.org/gnu/groff/groff-1.22.4.tar.gz*
MD5 sum: `08fb04335e2f5e73f23ea4c3adbf0c5f`

• GRUB (2.06) - 6,428 KB:
Home page: *https://www.gnu.org/software/grub/*
Download: *https://ftp.gnu.org/gnu/grub/grub-2.06.tar.xz*
MD5 sum: `cf0fd928b1e5479c8108ee52cb114363`

• Gzip (1.11) - 786 KB:
Home page: *https://www.gnu.org/software/gzip/*
Download: *https://ftp.gnu.org/gnu/gzip/gzip-1.11.tar.xz*
MD5 sum: `d1e93996dba00cab0caa7903cd01d454`

• Iana-Etc (20220207) - 580 KB:
Home page: *https://www.iana.org/protocols*
Download: *https://github.com/Mic92/iana-etc/releases/download/20220207/iana-etc-20220207.tar.gz*
MD5 sum: `81d865ce7fe4240d5abed48c3ca5fa9f`

• Inetutils (2.2) - 1,494 KB:
Home page: *https://www.gnu.org/software/inetutils/*
Download: *https://ftp.gnu.org/gnu/inetutils/inetutils-2.2.tar.xz*
MD5 sum: `de8c1b49cbde2b30e481c61c65357ad4`
SHA256 sum: `01b9a4bc73a47e63f6e8a07b76122d9ad2a2e46ebf14870e9c91d660b5647a22`

• Intltool (0.51.0) - 159 KB:
Home page: *https://freedesktop.org/wiki/Software/intltool*
Download: *https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz*
MD5 sum: `12e517cac2b57a0121cda351570f1e63`

• IPRoute2 (5.16.0) - 843 KB:
Home page: *https://www.kernel.org/pub/linux/utils/net/iproute2/*
Download: *https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.16.0.tar.xz*
MD5 sum: `994c1bad2a24aa9d70e89670c5b5dfcb`

• Kbd (2.4.0) - 1,095 KB:
Home page: *https://kbd-project.org/*
Download: *https://www.kernel.org/pub/linux/utils/kbd/kbd-2.4.0.tar.xz*
MD5 sum: `3cac5be0096fcf7b32dcbd3c53831380`

• Kmod (29) - 548 KB:
Download: *https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-29.tar.xz*
MD5 sum: `e81e63acd80697d001c8d85c1acb38a0`

• Less (590) - 348 KB:
Home page: *https://www.greenwoodsoftware.com/less/*
Download: *https://www.greenwoodsoftware.com/less/less-590.tar.gz*
MD5 sum: `f029087448357812fba450091a1172ab`

• LFS-Bootscripts (20210608) - 32 KB:
Download: *https://www.linuxfromscratch.org/lfs/downloads/11.1/lfs-bootscripts-20210608.tar.xz*
MD5 sum: `0f51a074cc4faaff93b3c80e9ab27b0c`

• Libcap (2.63) - 171 KB:
Home page: *https://sites.google.com/site/fullycapable/*
Download: *https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.63.tar.xz*
MD5 sum: `18410cec436f827e698ee9ea16ada9b7`

• Libffi (3.4.2) - 1,320 KB:
Home page: *https://sourceware.org/libffi/*
Download: *https://github.com/libffi/libffi/releases/download/v3.4.2/libffi-3.4.2.tar.gz*
MD5 sum: `294b921e6cf9ab0fbaea4b639f8fdbe8`

• Libpipeline (1.5.5) - 934 KB:
Home page: *http://libpipeline.nongnu.org/*
Download: *https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.5.tar.gz*
MD5 sum: `3e725c76bfea1985e87e851ee50c2e29`

• Libtool (2.4.6) - 951 KB:
Home page: *https://www.gnu.org/software/libtool/*
Download: *https://ftp.gnu.org/gnu/libtool/libtool-2.4.6.tar.xz*
MD5 sum: `1bfb9b923f2c1339b4d2ce1807064aa5`

• Linux (5.16.9) - 124,577 KB:
Home page: *https://www.kernel.org/*
Download: *https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.16.9.tar.xz*
MD5 sum: `4d6a704bf3e249ef6189b6f17457084b`

Nota

O kernel Linux é atualizado com relativa frequência, muitas vezes devido às descobertas de vulnerabilidades de segurança. A versão estável de kernel mais atual disponível pode ser usada, a menos que a página de errata diga o contrário.
Para usuárias(os) com largura de banda de velocidade limitada ou cara que desejem atualizar o kernel Linux, uma versão básica do pacote e patches pode ser baixada separadamente. Isso talvez economize algum tempo ou custo para uma posterior atualização de nível de patch contendo um lançamento menor.

• M4 (1.4.19) - 1,617 KB:
Home page: *https://www.gnu.org/software/m4/*
Download: *https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz*
MD5 sum: `0d90823e1426f1da2fd872df0311298d`

• Make (4.3) - 2,263 KB:
Home page: *https://www.gnu.org/software/make/*
Download: *https://ftp.gnu.org/gnu/make/make-4.3.tar.gz*
MD5 sum: `fc7a67ea86ace13195b0bce683fd4469`

• Man-DB (2.10.1) - 1,847 KB:
Home page: *https://www.nongnu.org/man-db/*
Download: *https://download.savannah.gnu.org/releases/man-db/man-db-2.10.1.tar.xz*
MD5 sum: `b03b76a9a00d0d6b2299b823fba4f579`

• Man-pages (5.13) - 1,752 KB:
Home page: *https://www.kernel.org/doc/man-pages/*
Download: *https://www.kernel.org/pub/linux/docs/man-pages/man-pages-5.13.tar.xz*
MD5 sum: `3ac24e8c6fae26b801cb87ceb63c0a30`

• Meson (0.61.1) - 1,963 KB:
Home page: *https://mesonbuild.com*
Download: *https://github.com/mesonbuild/meson/releases/download/0.61.1/meson-0.61.1.tar.gz*
MD5 sum: `8ed66d5537275df3defffb66d1fb897f`

• MPC (1.2.1) - 820 KB:
Home page: *http://www.multiprecision.org/*
Download: *https://ftp.gnu.org/gnu/mpc/mpc-1.2.1.tar.gz*
MD5 sum: `9f16c976c25bb0f76b50be749cd7a3a8`

• MPFR (4.1.0) - 1,490 KB:
Home page: *https://www.mpfr.org/*
Download: *https://www.mpfr.org/mpfr-4.1.0/mpfr-4.1.0.tar.xz*
MD5 sum: `bdd3d5efba9c17da8d83a35ec552baef`

• Ncurses (6.3) - 3,500 KB:
Home page: *https://www.gnu.org/software/ncurses/*
Download: *https://invisible-mirror.net/archives/ncurses/ncurses-6.3.tar.gz*
MD5 sum: `a2736befde5fee7d2b7eb45eb281cdbe`

• Ninja (1.10.2) - 209 KB:
Home page: *https://ninja-build.org/*
Download: *https://github.com/ninja-build/ninja/archive/v1.10.2/ninja-1.10.2.tar.gz*
MD5 sum: `639f75bc2e3b19ab893eaf2c810d4eb4`

• OpenSSL (3.0.1) - 14,660 KB:
Home page: *https://www.openssl.org/*
Download: *https://www.openssl.org/source/openssl-3.0.1.tar.gz*
MD5 sum: `7d07e849d77d276891edd579a8832bb3`

• Patch (2.7.6) - 766 KB:
Home page: *https://savannah.gnu.org/projects/patch/*
Download: *https://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz*
MD5 sum: `78ad9937e4caadcba1526ef1853730d5`

• Perl (5.34.0) - 12,580 KB:
Home page: *https://www.perl.org/*
Download: *https://www.cpan.org/src/5.0/perl-5.34.0.tar.xz*
MD5 sum: `df7ecb0653440b26dc951ad9dbfab517`

• Pkg-config (0.29.2) - 1,970 KB:
Home page: *https://www.freedesktop.org/wiki/Software/pkg-config*
Download: *https://pkg-config.freedesktop.org/releases/pkg-config-0.29.2.tar.gz*
MD5 sum: `f6e931e319531b736fadc017f470e68a`

• Procps (3.3.17) - 985 KB:
Home page: *https://sourceforge.net/projects/procps-ng*
Download: *https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.17.tar.xz*
MD5 sum: `d60613e88c2f442ebd462b5a75313d56`

• Psmisc (23.4) - 362 KB:
Home page: *https://gitlab.com/psmisc/psmisc*
Download: *https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.4.tar.xz*
MD5 sum: `8114cd4489b95308efe2509c3a406bbf`

• Python (3.10.2) - 18,341 KB:
Home page: *https://www.python.org/*
Download: *https://www.python.org/ftp/python/3.10.2/Python-3.10.2.tar.xz*
MD5 sum: `14e8c22458ed7779a1957b26cde01db9`

• Documentação de Python (3.10.2) - 7,102 KB:
Download: *https://www.python.org/ftp/python/doc/3.10.2/python-3.10.2-docs-html.tar.bz2*
MD5 sum: `ffa52f0017baf72df9d32dec785fd6ab`

• Readline (8.1.2) - 2,923 KB:
Home page: *https://tiswww.case.edu/php/chet/readline/rltop.html*
Download: *https://ftp.gnu.org/gnu/readline/readline-8.1.2.tar.gz*
MD5 sum: `12819fa739a78a6172400f399ab34f81`

• Sed (4.8) - 1,317 KB:
Home page: *https://www.gnu.org/software/sed/*
Download: *https://ftp.gnu.org/gnu/sed/sed-4.8.tar.xz*
MD5 sum: `6d906edfdb3202304059233f51f9a71d`

• Shadow (4.11.1) - 1,618 KB:
Home page: *https://shadow-maint.github.io/shadow/*
Download: *https://github.com/shadow-maint/shadow/releases/download/v4.11.1/shadow-4.11.1.tar.xz*
MD5 sum: `5a95ec069aa91508167d02fecafaa912`

• Sysklogd (1.5.1) - 88 KB:
Home page: *https://www.infodrom.org/projects/sysklogd/*
Download: *https://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.1.tar.gz*
MD5 sum: `c70599ab0d037fde724f7210c2c8d7f8`

• Sysvinit (3.01) - 124 KB:
Home page: *https://savannah.nongnu.org/projects/sysvinit*
Download: *https://download.savannah.gnu.org/releases/sysvinit/sysvinit-3.01.tar.xz*
MD5 sum: `dc14f92af715bcfa33cc25341730452e`

• Tar (1.34) - 2,174 KB:
Home page: *https://www.gnu.org/software/tar/*
Download: *https://ftp.gnu.org/gnu/tar/tar-1.34.tar.xz*
MD5 sum: `9a08d29a9ac4727130b5708347c0f5cf`

• Tcl (8.6.12) - 10,112 KB:
Home page: *http://tcl.sourceforge.net/*
Download: *https://downloads.sourceforge.net/tcl/tcl8.6.12-src.tar.gz*
MD5 sum: `87ea890821d2221f2ab5157bc5eb885f`

• Documentação de Tcl (8.6.12) - 1,176 KB:
Download: *https://downloads.sourceforge.net/tcl/tcl8.6.12-html.tar.gz*
MD5 sum: `a0d1a5b60bbb68f2f0bd3066a19c527a`

• Texinfo (6.8) - 4,848 KB:
Home page: *https://www.gnu.org/software/texinfo/*
Download: *https://ftp.gnu.org/gnu/texinfo/texinfo-6.8.tar.xz*
MD5 sum: `a91b404e30561a5df803e6eb3a53be71`

• Dados de Zona de Tempo (2021e) - 413 KB:
Home page: *https://www.iana.org/time-zones*
Download: *https://www.iana.org/time-zones/repository/releases/tzdata2021e.tar.gz*
MD5 sum: `4fdfad906ebc85fef30221c10964cce9`

• Udev-lfs Tarball (udev-lfs-20171102) - 11 KB:
Download: *https://anduin.linuxfromscratch.org/LFS/udev-lfs-20171102.tar.xz*
MD5 sum: `27cd82f9a61422e186b9d6759ddf1634`

• Util-linux (2.37.4) - 5,971 KB:
Home page: *https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/*
Download: *https://www.kernel.org/pub/linux/utils/util-linux/v2.37/util-linux-2.37.4.tar.xz*
MD5 sum: `755919e658c349cad9e1c7c771742d48`

• Vim (8.2.4383) - 15,622 KB:
Home page: *https://www.vim.org*
Download: *https://anduin.linuxfromscratch.org/LFS/vim-8.2.4383.tar.gz*
MD5 sum: `3168ff48e382a1201bd0cbd0209bd3e0`

Nota

A versão de vim muda diariamente. Para conseguir a versão mais atual, vá para *https://github.com/vim/vim/tags*.

• XML::Parser (2.46) - 249 KB:
Home page: *https://github.com/chorny/XML-Parser*
Download: *https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.46.tar.gz*
MD5 sum: `80bb18a8e6240fcf7ec2f7b57601c170`

• Xz Utils (5.2.5) - 1,122 KB:
Home page: *https://tukaani.org/xz*
Download: *https://tukaani.org/xz/xz-5.2.5.tar.xz*
MD5 sum: `aa1621ec7013a19abab52a8aff04fe5b`

• **Zlib (1.2.11) - 457 KB:**
Home page: *https://www.zlib.net/*
Download: *https://zlib.net/zlib-1.2.11.tar.xz*
MD5 sum: `85adef240c5f370b308da8c938951a68`

• **Zstd (1.5.2) - 1,892 KB:**
Home page: *https://facebook.github.io/zstd/*
Download: *https://github.com/facebook/zstd/releases/download/v1.5.2/zstd-1.5.2.tar.gz*
MD5 sum: `072b10f71f5820c24761a65f31f43e73`

Tamanho total desses pacotes: cerca de 446 MB

3.3. Patches Necessários

Em adição aos pacotes, vários patches também são exigidos. Esses patches corrigem quaisquer erros nos pacotes que deveriam ser consertados pela(o) Mantenedora(or). Os patches também fazem pequenas modificações para tornar os pacotes mais fáceis de se trabalhar. Os seguintes patches serão necessários para construir um sistema LFS:

• **Binutils LTO Fix Patch - 3.5 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/binutils-2.38-lto_fix-1.patch*
MD5 sum: `3df11b6123d5bbdb0fc83862a003827a`

• **Patch de Documentação de Bzip2 - 1.6 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/bzip2-1.0.8-install_docs-1.patch*
MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`

• **Patch de Correções de Internacionalização de Coreutils - 166 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/coreutils-9.0-i18n-1.patch*
MD5 sum: `1eeba2736dfea013509f9975365e4e32`

• **Patch de Correção de Chmod de Coreutils - 3.8 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/coreutils-9.0-chmod_fix-1.patch*
MD5 sum: `4709df88e68279e6ef357aa819ba5b1a`

• **Glibc FHS Patch - 2.8 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/glibc-2.35-fhs-1.patch*
MD5 sum: `9a5997c3452909b1769918c759eff8a2`

• **Patch de Correção de Backspace/Delete de Kbd - 12 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/kbd-2.4.0-backspace-1.patch*
MD5 sum: `f75cca16a38da6caa7d52151f7136895`

• **Patch de Correção de Desenvolvedora(or) de Perl - 1.6 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/perl-5.34.0-upstream_fixes-1.patch*
MD5 sum: `fb42558b59ed95ee00eb9f1c1c9b8056`

• **Patch Consolidado de Sysvinit - 2.4 KB:**
Download: *https://www.linuxfromscratch.org/patches/lfs/11.1/sysvinit-3.01-consolidated-1.patch*
MD5 sum: `4900322141d493e74020c9cf437b2cdc`

Tamanho total desses patches: cerca de 193.7 KB

Em adição aos patches exigidos acima, existe um número de patches opcionais criados pela comunidade LFS. Esses patches opcionais solucionam problemas menores ou habilitam funcionalidade que não está habilitada por padrão. Sinta-se à vontade para examinar o banco de dados de patches localizado em *https://www.linuxfromscratch.org/patches/downloads/* e adquirir quaisquer patches 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 construção o sistema temporário. Nós criaremos um conjunto de diretórios em `$LFS` para a instalação das ferramentas temporárias; adicionaremos uma(m) usuária(o) desprivilegiada(o) para reduzir risco; e criaremos um ambiente apropriado de construção para aquela(e) usuária(o). Nós também explicaremos a unidade de tempo que usamos para medir quanto tempo pacotes de LFS levam para construir, ou "SBUs", e daremos alguma informação acerca de suítes de teste de pacote.

4.2. Criando um layout limitado de diretório em sistema de arquivos de LFS

A primeira tarefa realizada na partição LFS é a de criar uma hierarquia limitada de diretório de forma que aplicativos compilados em Capítulo 6 (bem como glibc e libstdc++ em Capítulo 5) possam ser instalados no local final deles. Isso é necessário de maneira que aqueles aplicativos temporários sejam sobrescritos quando reconstruí-los em Capítulo 8.

Crie o layout exigido de diretório executando o seguinte 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
```

Aplicativos em Capítulo 6 serão compilados com um compilador cruzado (mais detalhes na seção Notas Técnicas do Conjunto de Ferramentas). Com a finalidade de separar esse compilador cruzado de outros aplicativos, ele será instalado em um diretório especial. Crie esse diretório com:

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

4.3. Adicionando a(o) Usuária(o) 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 facilitar a configuração de um ambiente de trabalho limpo, crie uma(m) nova(o) usuária(o) chamada(o) `lfs` como um membro de um novo grupo (também chamado `lfs`) e use essa(e) usuária(o) durante o processo de instalação. Como `root`, emita os seguintes comandos para adicionar a(o) nova(o) usuária(o):

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

O significado das opções de linha de comando:

`-s /bin/bash`
Isso torna **bash** o interpretador de comandos padrão para a(o) usuária(o) `lfs`.

-g lfs
Essa opção adiciona usuária(o) `lfs` ao grupo `lfs`.

-m
Isso cria um diretório home para `lfs`.

-k /dev/null
Esse parâmetro previne possível cópia de arquivos a partir de um diretório esqueleto (padrão é `/etc/skel`) mudando a localização de entrada para o dispositivo especial null.

lfs
Esse é o nome atual para a(o) usuária(o) criada(o).

Para se logar como `lfs` (em oposição a mudar para a(o) usuária(o) `lfs` quando logada(o) como `root`, que não exige que a(o) usuária(o) `lfs` tenha uma senha), dê a `lfs` uma 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
```

Se um diretório de trabalho separado foi criado como sugerido, então dê à(ao) usuária(o) `lfs` a propriedade desse diretório:

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

Nota

Em alguns sistemas anfitrião, o seguinte comando não completa adequadamente e suspende o login para a(o) usuária(o) lfs para o segundo plano. Se o prompt "lfs:~$" não aparecer imediatamente, então emitir o comando **fg** corrigirá o problema.

Em seguida, logue-se como usuária(o) `lfs`. Isso pode ser feito via um console virtual; por intermédio de um gerenciador de tela; ou com o seguinte comando de substituir/comutar 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 pode ser encontrada 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`, o shell inicial é geralmente 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 técnica usada aqui alcança o objetivo de assegurar um ambiente limpo.

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 de 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. Desativando a função hash, o shell sempre vai procurar no PATH quando 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 da(o) usuária(o) (umask) para 022 garante que recém criados arquivos e diretórios são graváveis somente por suas(eus) donas(os), mas são legíveis e executáveis por qualquer pessoa (assumindo que os modos padrão são usados pelas chamadas 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 dentro do ambiente chroot.

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 está contida em Notas Técnicas do Conjunto de Ferramentas.

PATH=/usr/bin

Muitas distribuições modernas de Linux mesclaram /bin e /usr/bin. Quando esse for o caso, a variável PATH padrão apenas precisa ser configurada para /usr/bin/ para o ambiente de Capítulo 6. Quando esse não for o caso, a seguinte linha adiciona /bin ao caminho.

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

Se `/bin` não for um link simbólico, então ele tem de ser adicionado 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 de Capítulo 5 é imediatamente pego pelo shell após sua instalação. 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`

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

`export ...`

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

Importante

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

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

Após o uso da(o) usuária(o) `lfs` for finalizado no início de Capítulo 7, você pode restaurar `/etc/bash.bashrc` (se desejado).

Perceba que o pacote Bash de LFS que nós construiremos em Seção 8.34, “Bash-5.1.16” não é configurado para carregar ou executar `/etc/bash.bashrc`, de modo que esse arquivo é inútil em um sistema LFS completo.

Finalmente, para ter o ambiente totalmente preparado para construção das ferramentas temporárias, carregue o recém criado perfil de usuária(o):

```
source ~/.bash_profile
```

4.5. Sobre UPCs

Muitas pessoas gostariam de saber de antemão aproximadamente quanto tempo leva para compilar e instalar cada pacote. Devido a Linux From Scratch poder ser construído em muitos sistemas, é impossível fornecer estimativas de tempo precisas. O maior pacote (Glibc) levará aproximadamente vinte (20) minutos em sistemas mais rápidos, mas poderia levar até três (03) dias em 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 neste livro é binutils em Capítulo 5. O tempo necessário para compilar esse pacote é que será referenciado 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 quatro e meio (4,5) UPCs. Isso significa que, se um sistema precisou de dez (10) minutos para compilar e instalar a primeira passagem de binutils, então será necessário *aproximadamente* quarenta e cinco (45) minutos para construir esse pacote de exemplo. Felizmente, a maioria dos tempos de construção é menor que o tempo para binutils.

Em geral, UPCs não são totalmente precisas, pois dependem de muitos fatores, incluindo a versão de 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.

Nota

Para muitos sistemas modernos com múltiplos processadores (ou cores) o tempo de compilação para um pacote pode ser reduzido realizando um "parallel make", seja configurando uma variável de ambiente; ou dizendo para o aplicativo **make** quantos processadores estão disponíveis. Por exemplo, uma CPU Intel i5-6500 pode suportar quatro processos simultâneos com:

```
export MAKEFLAGS='-j4'
```

ou somente construindo com:

```
make -j4
```

Quando múltiplos processadores são usados dessa maneira, as unidades UPC no livro irão variar ainda mais do que normalmente aconteceria. Em alguns casos, o passo make simplesmente falhará. Analisar a saída dos processos de construção também será mais difícil, pois as linhas 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.

4.6. Sobre as Suítes de Teste

A maioria dos pacotes fornece uma suíte de teste. Rodar 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 executa seu conjunto de verificações geralmente prova que o pacote está funcionando como a(o) desenvolvedora(r) pretendia. Entretanto isso não garante 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 a seu papel central em um sistema que funcione adequadamente. As suítes de teste para GCC e glibc podem levar bastante tempo para completarem, especialmente em uma máquina lenta, mas são fortemente recomendadas.

Nota

Executar as suítes de teste em Capítulo 5 e Capítulo 6 é impossível, dado que os aplicativos são compilados com um compilador cruzado, de forma que não se supõe que sejam aptos a executar no anfitrião de construção.

Um problema comum com a execução de suítes de teste para binutils e GCC é ficar sem 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 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, mas 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/11.1/* para verificar quando essas falhas são esperadas ou não. Esse site é válido para todos os testes ao longo deste livro.

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

Material Preliminar Importante

Introdução

Esta parte é dividida em três estágios: primeiro construindo 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

Com esta parte inicia o trabalho real de construir um novo sistema. Exige muito cuidado em assegurar que as instruções sejam seguidas exatamente conforme o livro as mostra. Você deveria tentar entender o que elas fazem, e qualquer que seja sua ânsia para finalizar sua construção, você deveria evitar digitá-las cegamente como mostrado, mas ler documentação quando houver algo que você não entenda. Além disso, acompanhe sua digitação e da saída de comandos, enviando-as para um arquivo, usando o utilitário **tee**. Isso permite um melhor diagnóstico se algo der errado.

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

Notas Técnicas do Conjunto de Ferramentas

Esta seção explana algumas das razões e detalhes técnicos por trás do método completo de construção. Não é essencial entender imediatamente tudo nesta seção. A maior parte desta informação ficará mais clara após executar uma construção atual. Esta seção pode e deve ser consultada a qualquer momento durante o processo.

O objetivo geral do Capítulo 5 e do Capítulo 6 é o de produzir uma área temporária que contém um conjunto reconhecidamente bom de ferramentas que pode ser isolado do sistema anfitrião. Usando-se **chroot**, os comandos nos capítulos subsequentes estarão confinados naquele ambiente, assegurando uma construção limpa e livre de problemas do sistema LFS alvo. O processo de construção foi desenhado para minimizar os riscos para leitores novatos e para prover o maior valor educacional ao mesmo tempo.

O processo de construção é baseado no processo de *compilação cruzada*. A compilação cruzada normalmente é usada para construir um compilador e o conjunto de ferramentas dele para uma máquina diferente daquela que é usada para a construção. Isso não é estritamente necessário para LFS, dado que a máquina onde o novo sistema executará é a mesma usada para a construção. Porém, a compilação cruzada tem a grande vantagem de que tudo o que é 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 o qual será usado para algum outro propósito que não construir 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 pode ser omitida em uma primeira leitura, retornar até ela mais tarde será benéfico para o seu completo entendimento do processo.

Permita-nos primeiro definir alguns termos usados nesse contexto

build
: é a máquina onde nós construímos aplicativos. Note que essa máquina é referenciada como sendo a “anfitriã” em outras seções.

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

target
: é usado apenas para compiladores. Ele é a máquina para a qual o compilador produz código. Ele pode ser diferente de ambos build e host.

Como um exemplo, permita-nos imaginar o seguinte cenário (as vezes rotulado de “Cruzado Canadense”): nós podemos ter um compilador somente em uma máquina lenta, vamos rotulá-la de “máquina A”, e o compilador de “ccA”. Nós também podemos ter uma máquina rápida (B), porém sem compilador, e nós eventualmente desejamos produzir código para outra máquina lenta (C). Para construir um compilador para a máquina “C”, nós teríamos três estágios:

Estágio	Build	Host	Target	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 outros programas necessários para a máquina C podem ser compilados usando cc2 na rápida máquina B. Note que a menos que B possa executar aplicativos produzidos por C, não existe maneira de testar os aplicativos construídos até que a própria máquina C esteja em execução. Por exemplo, para testar ccC, nós eventualmente desejamos adicionar um quarto estágio:

Estágio	Build	Host	Target	Ação
4	C	C	C	reconstruir e testar ccC usando o próprio 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 sendo executados. Tais compiladores são rotulados de compiladores *nativos*.

Implementação de Compilação Cruzada para LFS

Nota

Quase todos os sistemas de construção usam nomes da forma cpu-vendor-kernel-os rotulados como o trio de máquina. Um leitor atento eventualmente questionará porque um “trio” rotula um nome de quatro componentes. A razão é histórica: inicialmente, três nomes de componente eram suficientes para designar uma máquina inequivocamente, porém com novas máquinas e sistemas aparecendo, isso se provou insuficiente. A palavra “trio” subsistiu. Uma maneira simples de determinar seu trio de máquina é executar o script **config.guess** que vem com o fonte para muitos pacotes. Desempacote os fontes do pacote “binutils” e execute o script: **`./config.guess`** e observe a saída. Por exemplo, para um processador Intel de 32-bits, a saída será *i686-pc-linux-gnu*. Em um sistema de 64-bits, a saída será *x86_64-pc-linux-gnu*.

Esteja também ciente do nome do vinculador dinâmico da plataforma, frequentemente rotulado de carregador dinâmico (não confundir com o vinculador padrão **ld** o qual é parte do pacote “binutils”). O vinculador dinâmico provido por 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` para sistemas 64-bits. Uma maneira infalível de determinar o nome do vinculador dinâmico é inspecionar uma biblioteca aleatória do sistema anfitrião executando: **`readelf -l <nome de binário> | grep interpreter`** e observando a saída. A referência oficial cobrindo todas as plataformas está no arquivo `shlib-versions` na raiz da árvore do fonte do Glibc.

Para falsificar uma compilação cruzada em LFS, o nome do trio do anfitrião é ligeiramente ajustado modificando-se o campo "vendor" na variável `LFS_TGT`. Nós também usamos a opção *`--with-sysroot`* quando da construção do vinculador dinâmico e do compilador cruzado para informá-los onde encontrar os necessários arquivos do anfitrião. Isso assegura que nenhum dos outros aplicativos construídos em Capítulo 6 pode se vincular a bibliotecas na máquina de construção. Somente dois estágios são obrigatórios, e mais um para testes:

Estágio	Build	Host	Target	Ação
1	pc	pc	lfs	construir compilador cruzado cc1 usando

Estágio	Build	Host	Target	Ação
				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 o próprio em lfs

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

Agora, existe mais acerca de compilação cruzada: a linguagem C não é apenas um compilador, mas também define uma biblioteca padrão. Neste livro, a biblioteca GNU C, rotulada de “glibc”, é usada. Essa biblioteca deve ser compilada para a máquina lfs, isto é, usando o compilador cruzado cc1. Porém, o próprio compilador usa uma biblioteca interna implementando complexas instruções não disponíveis no conjunto de instruções do montador. Essa biblioteca interna é rotulada de “libgcc”, e deve ser vinculada à biblioteca “glibc” para ser completamente funcional! Além disso, a biblioteca padrão para C++ (libstdc++) também precisa estar vinculada à “glibc”. A solução para esse problema de ovo e galinha é primeiro construir uma libgcc degradada baseada em cc1, faltando algumas funcionalidades tais como camadas e manipulação de exceções, então construir glibc usando esse compilador degradado (o próprio glibc não degradado), então construir libstdc++. Porém, nessa última biblioteca faltarão as mesmas funcionalidades que libgcc.

Esse não é o fim da história: a conclusão do parágrafo precedente é a de que cc1 é incapaz de construir uma libstdc++ completamente funcional, porém esse é o único compilador disponível para construir as bibliotecas C/C++ durante o estágio 2! Certamente, o compilador construído durante o estágio 2, cc-lfs, seria capaz de construir aquelas bibliotecas, porém (1) o sistema de construção do GCC não sabe que está utilizável em pc; e (2) usá-lo em pc estaria sob o risco de vinculamento às bibliotecas de pc, dado que cc-lfs é um compilador nativo. Assim, nós temos de construir libstdc ++ mais tarde, em jaula.

Outros detalhes procedurais

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

Binutils é instalado primeiro, pois a execução de **configure** de ambos GCC e Glibc executa vários testes de características no montador e no vinculador para determinar quais características de software habilitar ou desabilitar. Isso é mais importante do que, inicialmente, alguém possa perceber. Um GCC ou 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 teste de suíte 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 uma localização são rigidamente vinculadas à outra. Uma faceta importante do vinculador é a ordem de procura de biblioteca dele. Informações detalhadas podem ser obtidas de **ld** passando a flag `--verbose`.

Por exemplo, **$LFS_TGT-ld --verbose | grep SEARCH** exibirá os caminhos atuais de procura e a ordem deles. Isso mostra quais arquivos estão vinculados por **ld** pela compilação de um aplicativo fictício e passagem do modificador *--verbose* ao vinculador. Por exemplo, **$LFS_TGT-gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded** exibirá todos os arquivos abertos com sucesso durante o vinculamento.

O próximo pacote instalado é o GCC. Um exemplo do que pode ser visto durante a execução dele de **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 busca os diretórios do PATH para encontrar quais ferramentas usar. Entretanto, durante a operação atual do próprio **gcc**, os mesmos caminhos de busca não são necessariamente usados. Para descobrir qual vinculador padrão **gcc** usará, execute: **$LFS_TGT-gcc -print-prog-name=ld**.

Informação detalhada pode ser obtida de **gcc** passando-se a opção de linha de comando *-v* durante a compilação de um aplicativo fictício. Por exemplo, **gcc -v dummy.c** exibirá informação detalhada acerca do preprocessador, compilação e estágios da montagem, incluindo os caminhos de busca incluídos do **gcc** e a ordem deles.

Os próximos instalados são os cabeçalhos sanitizados da API do Linux. Eles permitem a interface da biblioteca C padrão (Glibc) com características que o kernel Linux proverá.

O próximo pacote instalado é Glibc. As considerações mais importantes para a construção do Glibc são o compilador, ferramentas binárias e os cabeçalhos do kernel. O compilador geralmente não é um problema dado que Glibc sempre usará o compilador relacionado ao parâmetro *--host* passado ao script de configuração dele; por exemplo, em nosso caso, o compilador será **$LFS_TGT-gcc**. As ferramentas binárias e os cabeçalhos do kernel podem ser um bocado mais complicados. Dessa maneira, nós não nos arriscamos e usamos os modificadores de configuração disponíveis para forçar as seleções corretas. Após a execução de **configure**, verifique o conteúdo do arquivo `config.make` no diretório `build` para todos os detalhes importantes. Observe o uso de *CC="$LFS_TGT-gcc"* (com `$LFS_TGT` expandida) para controlar quais ferramentas binárias são usadas e o uso das flags *-nostdinc* e *-isystem* para controlar o caminho de busca incluído do compilador. Esses itens destacam um importante aspecto do pacote Glibc—ele é muito autossuficiente em termos de maquinário de construção e geralmente não confia em padrões de conjuntos de ferramentas.

Como dito acima, a biblioteca C++ padrão é compilada depois, seguida em Capítulo 6 por todos os aplicativos que necessitam deles próprios para serem construídos. O passo de instalação de todos aqueles pacotes usa os aplicativos instalados no sistema de arquivos do LFS.

Ao final do Capítulo 6 o compilador nativo do LFS é instalado. Primeiro binutils-pass2 é construído, com a mesma instalação `DESTDIR` como os outros programas, então a segunda passagem de GCC é construída, omitindo libstdc++ e outras bibliotecas não importantes. Devido a algumas lógicas estranhas no script configure do GCC, `CC_FOR_TARGET` termina como **cc** quando o host for o mesmo que o target, porém for diferente do sistema de construção. Essa é a razão pela qual *CC_FOR_TARGET=$LFS_TGT-gcc* é colocado explicitamente nas opções de configuração.

Uma vez dentro do ambiente chroot no Capítulo 7, a primeira tarefa é instalar libstdc++. Então instalações temporárias de programas necessários para a operação apropriada do conjunto de ferramentas são executadas. Deste ponto em diante, o conjunto central de ferramentas está autocontido e auto-hospedado. No Capítulo 8, as versões finais de todos os pacotes necessários para um sistema completamente funcional são construídos, testados e instalados.

Instruções Gerais de Compilação

Quando da construção de pacotes existem várias suposições feitas dentro das instruções:

- Vários dos pacotes recebem patches antes da compilação, mas apenas quando o patch é necessário para evitar um problema. Um patch frequentemente é necessário tanto neste quanto nos seguintes capítulos, mas algumas vezes em apenas uma localização. Portanto, não se preocupe se as instruções para um patch baixado pareçam estar faltando. Mensagens de alerta acerca de *offset* ou *fuzz* também talvez sejam encontradas quando da aplicação de um patch. Não se preocupe com esses alertas, uma vez que o patch ainda foi aplicado com sucesso.
- Durante a compilação da maior parte dos pacotes, existirão vários alertas que rolarão na tela. Esses são normais e seguramente podem ser ignorados. Esses alertas são o que parecem—alertas acerca de uso de sintaxe C ou C++ obsoleta, mas não inválida. Padrões C mudam com ampla frequência, e alguns pacotes ainda usam o padrão antigo. Isso não é um problema, mas gera o alerta.
- Verifique uma última vez que a variável de ambiente `LFS` está configurada adequadamente:

```
echo $LFS
```

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

- Finalmente, dois itens importantes devem ser enfatizados:

Importante

As instruções de construção assumem que as Exigências do Sistema Anfitrião, incluindo links simbólicos, foram 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

Para reenfatizar o processo de construção:

1. Coloque todos os pacotes e patches em um diretório que estará acessível a partir do ambiente chroot, tal como `/mnt/lfs/sources/`.
2. Mude para o diretório dos fontes.
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 a(o) usuária(o) *lfs* quando extrair o pacote.

      Todos os métodos para obter a árvore de código fonte sendo construído em-posição, exceto extrair o tarball de pacote, não são suportados. Notadamente, usar **cp -R** para copiar a árvore de código fonte para outro lugar pode destruir links e marcas temporais na árvore de fontes e causar falha de construção.
   b. Mude para o diretório criado quando o pacote foi extraído.
   c. Siga as instruções do livro para construir o pacote.
   d. Mude de volta para o diretório de fontes.
   e. Delete o diretório de fonte extraído a menos que instruído o contrário.

Capítulo 5. Compilando um Conjunto de Ferramentas Cruzado

5.1. Introdução

Este capítulo mostra como construir um compilador cruzado e suas ferramentas associadas. 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 em seus locais finais, dado que elas pertencem ao sistema que queremos construir.

5.2. Binutils-2.38 - 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: 620 MB

5.2.1. Instalação de Binutils Cruzado

Nota

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

É importante que Binutils seja o primeiro pacote compilado, pois ambos Glibc e GCC realizam vários testes sobre o vinculador e montador disponíveis para determinar quais de suas próprias características habilitar.

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

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

Nota

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

Agora prepare Binutils para compilação:

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

O significado das opções do configure:

`--prefix=$LFS/tools`
Isso diz para o script configure para preparar para instalar os aplicativos de binutils no diretório `$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.

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

Continue compilando o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados em Seção 8.18.2, “Conteúdo de Binutils.”

5.3. GCC-11.2.0 - Passagem 1

O pacote GCC contém a GNU compiler collection, o qual inclui os compiladores C e C++.

Tempo aproximado de construção: 11 UPC

Espaço em disco exigido: 3,3 GB

5.3.1. Instalação de GCC Cruzado

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 GCC. Desempacote cada pacote dentro do diretório de fonte de GCC e renomeie os diretórios resultantes de forma que os procedimentos de construção de GCC automaticamente os usarão:

Nota

Existem mal-entendidos frequentes sobre este capítulo. Os procedimentos são os mesmos que todos os outros capítulos explicados anteriormente (Instruções de construção de pacote). Primeiro extraia o tarball de gcc a partir do diretório de fontes e então mude para o diretório criado. Somente então deveria você prosseguir com as instruções abaixo.

```
tar -xf ../mpfr-4.1.0.tar.xz
mv -v mpfr-4.1.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 gmp
tar -xf ../mpc-1.2.1.tar.gz
mv -v mpc-1.2.1 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 de GCC recomenda construir GCC em um diretório de construção dedicado:

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

Prepare GCC para compilação:

```
../configure                  \
    --target=$LFS_TGT         \
    --prefix=$LFS/tools       \
    --with-glibc-version=2.35 \
    --with-sysroot=$LFS       \
    --with-newlib             \
    --without-headers         \
    --enable-initfini-array   \
    --disable-nls             \
    --disable-shared          \
    --disable-multilib        \
    --disable-decimal-float   \
    --disable-threads         \
    --disable-libatomic       \
    --disable-libgomp         \
    --disable-libquadmath     \
    --disable-libssp          \
    --disable-libvtv          \
    --disable-libstdcxx       \
    --enable-languages=c,c++
```

O significado das opções de configure:

`--with-glibc-version=2.35`
Essa opção especifica a versão de glibc a qual será usada no alvo. Ela não é relevante para a libc da distro anfitriã, pois tudo compilado por gcc passagem 1 executará no ambiente chroot, o qual é isolado de libc da distro 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 de libc.

`--without-headers`
Quando da criação de um compilador cruzado completo, GCC exige cabeçalhos padrão compatíveis com o sistema alvo. Para nossos propósitos esses cabeçalhos não serão necessários. Essa chave evita que GCC procure por eles.

`--enable-initfini-array`
Essa chave força o uso de algumas estruturas internas de dados que são necessárias, porém não podem ser detectadas quando da construção de um compilador cruzado.

`--disable-shared`
Essa chave força GCC a vincular suas bibliotecas internas estaticamente. Nós precisamos disso, pois as bibliotecas compartilhadas exigem glibc, que ainda não está instalado no sistema alvo.

`--disable-multilib`
Em x86_64, LFS não suporta uma configuração multilib. Essa chave é inofensiva para x86.

`--disable-decimal-float, --disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp, --disable-libvtv, --disable-libstdcxx`
Essas chaves desabilitam suporte para a extensão de ponto flutuante decimal, threading, libatomic, libgomp, libquadmath, libssp, libvtv, e a biblioteca padrão C++ respectivamente. Essas características falharão na

compilação quando da construção de um compilador cruzado e não são necessárias 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 GCC executando:

```
make
```

Instale o pacote:

```
make install
```

Essa construção de GCC instalou um par de cabeçalhos internos de sistema. Normalmente um deles, `limits.h`, sequencialmente incluiria o cabeçalho `limits.h` de sistema correspondente, nesse caso, `$LFS/usr/include/limits.h`. Entretanto, ao tempo dessa construção de GCC, `$LFS/usr/include/limits.h` não existe, de forma que o cabeçalho interno recém instalado é um arquivo parcial, autocontido, e não inclui as características estendidas do cabeçalho de sistema. Isso é adequado para a construção de glibc, porém o cabeçalho interno completo será necessário mais tarde. Crie uma versão completa do cabeçalho interno usando um comando que é idêntico ao que o sistema de construção de GCC faz em circunstâncias normais:

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

Detalhes acerca deste pacote estão localizados em Seção 8.26.2, “Conteúdo de GCC.”

5.4. Cabeçalhos da API do Linux-5.16.9

Os Cabeçalhos da API do Linux (em linux-5.16.9.tar.xz) expõem a API do kernel para uso por Glibc.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 1,2 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 em LFS) usar. Isso é feito por meio de sanitizar vários arquivos de cabeçalho C que são incluídos no tarball de fonte de kernel Linux.

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

```
make mrproper
```

Agora extraia os cabeçalhos de kernel visíveis ao usuário a partir do fonte. O alvo recomendado de make "headers_install" não pode ser usado, pois ele exige rsync, que talvez não esteja disponível. Os cabeçalhos são primeiro colocados em `./usr`, então copiados para a localização necessária.

```
make headers
find usr/include -name '.*' -delete
rm usr/include/Makefile
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, and /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, and /usr/include/xen

Descrições Curtas

`/usr/include/asm/*.h`	Os Cabeçalhos API ASM Linux
`/usr/include/asm-generic/*.h`	Os Cabeçalhos Genéricos API ASM Linux
`/usr/include/drm/*.h`	Os Cabeçalhos API DRM Linux
`/usr/include/linux/*.h`	Os Cabeçalhos Linux API Linux
`/usr/include/misc/*.h`	Os Cabeçalhos Miscelâneas API Linux
`/usr/include/mtd/*.h`	Os Cabeçalhos API MTD Linux
`/usr/include/rdma/*.h`	Os Cabeçalhos API RDMA Linux
`/usr/include/scsi/*.h`	Os Cabeçalhos API SCSI Linux
`/usr/include/sound/*.h`	Os Cabeçalhos de Som API Linux
`/usr/include/video/*.h`	Os Cabeçalhos de Vídeo API Linux

`/usr/include/xen/*.h`	Os Cabeçalhos Xen API Linux

5.5. Glibc-2.35

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

Tempo aproximado de construção: 4,3 UPC
Espaço em disco exigido: 818 MB

5.5.1. Instalação de Glibc

Primeiro, crie um link simbólico para conformidade com 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 umas poucas versões sintáticas, de forma que tenha certeza de verificar **info coreutils ln** e `ln(1)` antes de relatar o que você talvez pense que seja um erro.

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

```
patch -Np1 -i ../glibc-2.35-fhs-1.patch
```

A documentação de Glibc recomenda construir 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 Glibc para compilação:

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

O significado das opções de configure:

`--host=$LFS_TGT, --build=$(../scripts/config.guess)`
O efeito combinado dessas chaves é o de que o sistema de construção do Glibc se autoconfigura para ser compilado cruzadamente, usando o vinculador cruzado e compilador cruzado em `$LFS/tools`.

`--enable-kernel=3.2`
Isso diz a Glibc para compilar a biblioteca com suporte para kernels Linux 3.2 e posteriores. Contornos para kernels antigos não estão habilitados.

`--with-headers=$LFS/usr/include`
Isso diz a Glibc para compilar a si mesmo com os cabeçalhos recentemente instalados no diretório $LFS/usr/include, de forma que ele saiba exatamente quais características o kernel 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.

Durante este estágio o seguinte alerta 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 aplicativo **msgfmt** faltando ou incompatível geralmente é inofensivo. Esse aplicativo **msgfmt** é parte do pacote Gettext que a distribuição anfitriã deveria fornecer.

Nota

Tem havido relatos de que esse pacote talvez falhe quando da construção como um "parallel make". Se isso ocorrer, então reexecute o comando make com uma 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 possivelmente o tornará inutilizável. Portanto, verifique duas vezes se o ambiente está corretamente configurado, antes de executar o seguinte comando.

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

O significado da opção make install:

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

Corrija caminho codificado rigidamente para o carregador de executável em 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(){}' > dummy.c
$LFS_TGT-gcc dummy.c
readelf -l a.out | grep '/ld-linux'
```

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

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

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

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

Uma vez que tudo esteja bem, limpe os arquivos de teste:

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

Nota

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

Agora que nosso conjunto de ferramentas cruzadas está completa, finalize a instalação do cabeçalho limits.h. Para fazer isso, execute um utilitário fornecido pelas(os) desenvolvedoras(os) de GCC:

```
$LFS/tools/libexec/gcc/$LFS_TGT/11.2.0/install-tools/mkheaders
```

Detalhes acerca deste pacote estão localizados em Seção 8.5.3, “Conteúdo de Glibc.”

5.6. Libstdc++ oriundo de GCC-11.2.0, Passagem 1

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 sua instalação quando construímos gcc-pass1, pois ela depende de glibc, que ainda não estava disponível no diretório alvo.

Tempo aproximado de construção: 0,4 UPC
Espaço em disco exigido: 818 MB

5.6.1. Instalação de Libstdc++ Alvo

Nota

Libstdc++ é parte dos fontes de GCC. Você deveria primeiro desempacotar o tarball de GCC e mudar para o diretório `gcc-11.2.0`.

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

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

Prepare 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++/11.2.0
```

O significado das opções de 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 include pré-compilados, os quais não são necessários neste estágio.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/11.2.0`
Isso especifica o diretório de instalação para arquivos include. Por causa de libstdc++ ser a biblioteca padrão C++ para LFS, esse diretório deveria coincidir com a localização onde o compilador C++ (**$LFS_TGT-g++**) procuraria pelos arquivos include C++ padrão. Em uma construção normal, essa informação é automaticamente passada para as opções **configure** de libstdc++ a partir do diretório de nível de topo. Em nosso caso, essa informação deve ser explicitamente dada. O compilador C++ precederá o caminho raiz de sistema `$LFS` (especificado quando da construção de GCC passagem 1) para o caminho de pesquisa de arquivo include, de forma que ele atualmente pesquisará em `$LFS/tools/$LFS_TGT/include/c++/11.2.0`. A combinação da variável *`DESTDIR`* (no comando **make install** abaixo) e essa chave garante instalar os cabeçalhos lá.

Compile libstdc++ executando:

```
make
```

Instale a biblioteca:

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

Detalhes acerca deste pacote estão localizados em Seção 8.26.2, “Conteúdo de 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 cruzadas. 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.

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

Uma vez mais, permita-nos relembrar que a configuração inapropriada de `LFS` junto com a construção como `root`, talvez torne seu computador inutilizável. Este capítulo inteiro precisa ser feito como usuária(o) `lfs`, com o ambiente conforme descrito em Seção 4.4, “Configurando o Ambiente”.

6.2. M4-1.4.19

O pacote M4 contém um processador de macro.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 31 MB

6.2.1. Instalação de M4

Prepare 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 em Seção 8.12.2, “Conteúdo de M4.”

6.3. Ncurses-6.3

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

Tempo aproximado de construção: 0,7 UPC
Espaço em disco exigido: 50 MB

6.3.1. Instalação de 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 Ncurses para compilação:

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

O significado das novas opções de configure:

`--with-manpage-format=normal`
Isso evita que Ncurses instale páginas comprimidas de manual, o que talvez aconteceu se a própria distribuição anfitriã tiver páginas comprimidas de manual.

`--without-ada`
Isso assegura que Ncurses não construa suporte para o compilador Ada, o qual talvez 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 despojar os aplicativos usando o aplicativo **strip** oriundo do anfitrião. O uso de ferramentas de anfitrião em aplicativo compilado cruzadamente pode causar falha.

`--enable-widec`
Essa chave faz com que bibliotecas de caracteres largos (por exemplo, `libncursesw.so.6.3`) sejam construídas em vez de bibliotecas normais (por exemplo, `libncurses.so.6.3`). Essas bibliotecas de

caracteres largos são utilizáveis tanto em locales de múltiplos bytes quanto em tradicionais de oito (08) bits, enquanto bibliotecas normais funcionam adequadamente só em locales de oito (08) bits. Bibliotecas de caracteres largos e bibliotecas normais são compatíveis em fonte, mas não são compatíveis em binário.

`--without-normal`
Essa chave desabilita a construção e instalação da maioria das bibliotecas estáticas.

Compile o pacote:

```
make
```

Instale o pacote:

```
make DESTDIR=$LFS TIC_PATH=$(pwd)/build/progs/tic install
echo "INPUT(-lncursesw)" > $LFS/usr/lib/libncurses.so
```

O significado das opções de install:

`TIC_PATH=$(pwd)/build/progs/tic`
Nós precisamos passar o caminho do recém construído **tic** apto para executar na máquina de construção, de forma que a base de dados de terminal possa ser criada sem erros.

echo "INPUT(-lncursesw)" > $LFS/usr/lib/libncurses.so
A biblioteca `libncurses.so` é necessária para uns poucos pacotes que nós construiremos breve. Nós criamos esse pequeno script vinculador, pois isso é o que é feito em Capítulo 8.

Detalhes acerca deste pacote estão localizados em Seção 8.28.2, “Conteúdo de Ncurses.”

6.4. Bash-5.1.16

O pacote Bash contém o Bourne-Again SHell.

Tempo aproximado de construção: 0,4 UPC
Espaço em disco exigido: 64 MB

6.4.1. Instalação de Bash

Prepare Bash para compilação:

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

O significado das opções de 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, Bash usará as funções `malloc` originárias de 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 em Seção 8.34.2, “Conteúdo do Bash.”

6.5. Coreutils-9.0

O pacote Coreutils contém utilitários para mostrar e configurar as características básicas de sistema.

Tempo aproximado de construção: 0,6 UPC
Espaço em disco exigido: 158 MB

6.5.1. Instalação de Coreutils

Prepare 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 de 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 de Perl.

Compile o pacote:

```
make
```

Instale o pacote:

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

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

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

Detalhes acerca deste pacote estão localizados em Seção 8.53.2, “Conteúdo do Coreutils.”

6.6. Diffutils-3.8

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

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 27 MB

6.6.1. Instalação de Diffutils

Prepare Diffutils 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 em Seção 8.55.2, “Conteúdo do Diffutils.”

6.7. File-5.41

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

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 32 MB

6.7.1. Instalação de 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 construí-lo:

```
mkdir build
pushd build
  ../configure --disable-bzlib      \
               --disable-libseccomp \
               --disable-xzlib      \
               --disable-zlib
  make
popd
```

O significado da nova opção de configure:

`--disable-*`
O script de configuração tenta usar alguns pacotes originários da distribuição anfitriã se os arquivos de biblioteca correspondentes existirem. Isso talvez cause falha de compilação se um arquivo de biblioteca existir, porém os arquivos de cabeçalhos correspondentes não. Essas opções impedem o uso dessas capacidades desnecessárias a partir do anfitrião.

Prepare 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
```

Detalhes acerca deste pacote estão localizados em Seção 8.10.2, “Conteúdo de File.”

6.8. Findutils-4.9.0

O pacote Findutils contém aplicativos para procurar arquivos. Esses aplicativos são fornecidos para procurar recursivamente dentro de uma árvore de diretório e para criar, manter e buscar um banco de dados (geralmente mais rápido que o find recursivo, porém não é confiável se o banco de dados não for atualizado recentemente).

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 42 MB

6.8.1. Instalação de Findutils

Prepare 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 em Seção 8.57.2, “Conteúdo do Findutils.”

6.9. Gawk-5.1.1

O pacote Gawk contém aplicativos para manipular arquivos de texto.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 45 MB

6.9.1. Instalação de Gawk

Primeiro, garanta que alguns arquivos desnecessários não sejam instalados:

```
sed -i 's/extras//' Makefile.in
```

Prepare 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 em Seção 8.56.2, “Conteúdo do Gawk.”

6.10. Grep-3.7

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

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 26 MB

6.10.1. Instalação de Grep

Prepare Grep 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 em Seção 8.33.2, “Conteúdo do Grep.”

6.11. Gzip-1.11

O pacote Gzip contém aplicativos para compressão e descompressão de arquivos.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 11 MB

6.11.1. Instalação de Gzip

Prepare 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 em Seção 8.60.2, “Conteúdo do Gzip.”

6.12. Make-4.3

O pacote Make contém um aplicativo para controlar a geração de executáveis e outros arquivos não fonte de um pacote a partir de arquivos fonte.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 15 MB

6.12.1. Instalação de Make

Prepare Make para compilação:

```
./configure --prefix=/usr   \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

O significado da nova opção de configure:

`--without-guile`
Apesar de nós estarmos compilando cruzadamente, configure tenta usar guile a partir do anfitrião de construção se encontrá-lo. Isso provoca falha de compilação, de forma que essa chave impede o uso de guile.

Compile o pacote:

```
make
```

Instale o pacote:

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

Detalhes acerca deste pacote estão localizados em Seção 8.64.2, “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 “patch” tipicamente criado pelo aplicativo **diff**.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 12 MB

6.13.1. Instalação de Patch

Prepare 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 em Seção 8.65.2, “Conteúdo do Patch.”

6.14. Sed-4.8

O pacote Sed contém um editor de fluxo.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 20 MB

6.14.1. Instalação de Sed

Prepare Sed 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 em Seção 8.29.2, “Conteúdo do Sed.”

6.15. Tar-1.34

O pacote Tar fornece a habilidade para criar arquivamentos tar bem como realizar vários outros tipos de manipulação de arquivamento. Tar pode ser usado em arquivamentos previamente criados para extrair arquivos, para armazenar arquivos adicionais, ou para atualizar ou listar arquivos que já foram armazenados.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 38 MB

6.15.1. Instalação de Tar

Prepare 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 em Seção 8.66.2, “Conteúdo do Tar.”

6.16. Xz-5.2.5

O pacote Xz contém aplicativos para compressão e descompressão de arquivos. Ele fornece capacidades para os formatos de compressão lzma e o mais novo xz. Comprimir arquivos de texto com **xz** gera uma melhor percentagem de compressão que os tradicionais comandos **gzip** ou **bzip2**.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 15 MB

6.16.1. Instalação de Xz

Prepare Xz para compilação:

```
./configure --prefix=/usr                       \
            --host=$LFS_TGT                     \
            --build=$(build-aux/config.guess) \
            --disable-static                    \
            --docdir=/usr/share/doc/xz-5.2.5
```

Compile o pacote:

```
make
```

Instale o pacote:

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

Detalhes acerca deste pacote estão localizados em Seção 8.8.2, “Conteúdo de Xz.”

6.17. Binutils-2.38 - Passagem 2

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

Tempo aproximado de construção: 1,3 UPC
Espaço em disco exigido: 520 MB

6.17.1. Instalação de Binutils

Binutils entrega uma cópia desatualizada de libtool no tarball. Ela carece de suporte de raiz de sistema de forma que os binários produzidos serão erroneamente vinculados à 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 Binutils para compilação:

```
../configure                   \
    --prefix=/usr              \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --disable-nls              \
    --enable-shared            \
    --disable-werror           \
    --enable-64-bit-bfd
```

O significado das novas opções de configure:

`--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). Talvez não seja necessário em sistemas de 64 bits, porém não causa dano.

Compile o pacote:

```
make
```

Instale o pacote:

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

Detalhes acerca deste pacote estão localizados em Seção 8.18.2, “Conteúdo de Binutils.”

6.18. GCC-11.2.0 - Passagem 2

O pacote GCC contém a GNU compiler collection, o qual inclui os compiladores C e C++.

Tempo aproximado de construção: 11 UPC

Espaço em disco exigido: 3,3 GB

6.18.1. Instalação de GCC

Como na primeira construção de GCC, os pacotes GMP, MPFR, e MPC são exigidos. Desempacote os tarballs e mova-os para os nomes de diretório exigidos:

```
tar -xf ../mpfr-4.1.0.tar.xz
mv -v mpfr-4.1.0 mpfr
tar -xf ../gmp-6.2.1.tar.xz
mv -v gmp-6.2.1 gmp
tar -xf ../mpc-1.2.1.tar.gz
mv -v mpc-1.2.1 mpc
```

Se construindo 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
```

Crie um diretório de construção separado novamente:

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

Crie um link simbólico que permite que libgcc seja construída com suporte posix de camadas:

```
mkdir -pv $LFS_TGT/libgcc
ln -s ../../../libgcc/gthr-posix.h $LFS_TGT/libgcc/gthr-default.h
```

Antes de iniciar a construção de GCC, lembre-se de desconfigurar quaisquer variáveis de ambiente que substituam os sinalizadores de otimização padrão.

Agora prepare GCC para compilação:

```
../configure                                       \
    --build=$(../config.guess)                     \
    --host=$LFS_TGT                                \
    --prefix=/usr                                  \
    CC_FOR_TARGET=$LFS_TGT-gcc                     \
    --with-build-sysroot=$LFS                      \
    --enable-initfini-array                        \
    --disable-nls                                  \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libquadmath                          \
    --disable-libssp                               \
    --disable-libvtv                               \
    --disable-libstdcxx                            \
    --enable-languages=c,c++
```

O significado das novas opções de configure:

`-with-build-sysroot=$LFS`

Normalmente, usar *`--host`* garante que um compilador cruzado seja usado para construir GCC, e que o compilador sabe que tem que procurar por cabeçalhos e bibliotecas em `$LFS`. Porém, o sistema de construção de GCC usa outras ferramentas, que não estão cientes dessa localização. Essa chave é necessária para que elas busquem os arquivos necessários em `$LFS`, e não no anfitrião.

`--enable-initfini-array`

Essa opção é automaticamente habilitada quando da construção de um compilador nativo com um compilador nativo em x86. Porém, aqui, nós construímos com um compilador cruzado, de forma que nós precisamos explicitamente configurar essa opção.

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 C de GNU nem sempre está instalado. Executar **cc** deixa a(o) administradora(r) de sistema livre para decidir qual compilador C instalar:

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

Detalhes acerca deste pacote estão localizados em Seção 8.26.2, “Conteúdo de GCC.”

Capítulo 7. Entrando em Chroot e Construindo Ferramentas Temporárias Adicionais

7.1. Introdução

Este capítulo mostra como construir os últimos bits que faltam no sistema temporário: as ferramentas necessárias para o maquinário de construção de vários pacotes. Agora que todas as dependências circulares foram resolvidas, um ambiente "chroot", completamente isolado do sistema operacional anfitrião (exceto pelo kernel em execução), pode ser usado para a construção.

Para operação adequada do ambiente isolado, alguma comunicação com o kernel em execução precisa ser estabelecida. Isso é feito por meio dos assim chamados *Sistemas de Arquivos Virtuais de Kernel*, que precisam ser montados quando da entrada no ambiente chroot. Você talvez queira verificar que eles estejam montados emitindo **findmnt**.

Até Seção 7.4, "Entrando no Ambiente Chroot", os comandos precisam ser executados como `root`, com a variável `LFS` configurada. Após a entrada em chroot, todos os comandos são executados como root, por sorte sem acesso ao OS do computador no qual que você construiu LFS. Seja cuidadosa(o) 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 logada(o) como usuária(o) `root` e não mais como usuária(o) `lfs`. Também, verifique duplamente que `$LFS` está configurada no ambiente do `root`.

Atualmente, a hierarquia de diretório inteira em `$LFS` é de propriedade da(o) usuária(o) `lfs`, uma(m) usuária(o) que existe somente no sistema anfitrião. Se os diretórios e arquivos sob `$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 receber 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 -R root:root $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown -R root:root $LFS/lib64 ;;
esac
```

7.3. Preparando Sistemas de Arquivos Virtuais de Kernel

Vários sistemas de arquivos exportados pelo kernel são usados para comunicar para e oriunda do próprio kernel. Esses sistemas de arquivos são virtuais uma vez que nenhum espaço de disco é usado por eles. O conteúdo dos sistemas de arquivos reside em memória.

Comece criando diretórios nos quais os sistemas de arquivos serão montados:

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

7.3.1. Criando Nós de Dispositivos Iniciais

Quando o kernel inicializa o sistema, ele exige a presença de alguns nós de dispositivos, em particular os dispositivos `console` e `null`. Os nós de dispositivos precisam ser criados no disco rígido de modo que eles estejam disponíveis antes que o kernel povoe `/dev`), e adicionalmente quando Linux é iniciado com *`init=/bin/bash`*. Crie os dispositivos executando os seguintes comandos:

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

7.3.2. Montando e Povoando /dev

O método recomendado de povoar o diretório `/dev` com dispositivos é montar um sistema de arquivos virtuais (tal como `tmpfs`) no diretório `/dev`, e permitir que os dispositivos sejam criados dinamicamente naquele sistema de arquivos virtuais conforme eles sejam detectados ou acessados. Criação de dispositivos é geralmente feita durante o processo de inicialização por Udev. Uma vez que esse novo sistema ainda não tem Udev e ainda não foi inicializado, é necessário montar e povoar `/dev` manualmente. Isso é conseguido montando com bind o diretório `/dev` do sistema anfitrião. Uma montagem com bind é um tipo especial de montagem que permite que você crie um espelho de um diretório ou ponto de montagem para alguma outra localização. Use o seguinte comando para conseguir isso:

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

7.3.3. Montando Sistemas de Arquivos Virtuais de Kernel

Agora monte os restantes sistemas de arquivos virtuais de kernel:

```
mount -v --bind /dev/pts $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Em alguns sistemas anfitrião, `/dev/shm` é um link simbólico para `/run/shm`. O tmpfs /run foi montado acima então, nesse caso, apenas um diretório precisa ser criado.

```
if [ -h $LFS/dev/shm ]; then
  mkdir -pv $LFS/$(readlink $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 hora de entrar no ambiente chroot para finalizar a instalação das restantes ferramentas temporárias. Esse ambiente estará em uso também para a instalação do sistema final. Como usuária(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     \
    /bin/bash --login
```

A opção *-i* dada para o comando **env** limpará todas as variáveis do ambiente chroot. Depois disso, apenas as variáveis HOME, TERM, PS1, e PATH são configuradas novamente. A construção *TERM=$TERM* configurará a variável TERM dentro de chroot para o mesmo valor que fora de chroot. Essa variável é necessária para aplicativos como **vim** e **less** operarem adequadamente. Se outras variáveis forem desejadas, tais como CFLAGS ou CXXFLAGS, então esse é um bom lugar para configurá-las novamente.

Deste ponto em diante, não mais há necessidade de usar a variável LFS, pois todo o trabalho estará restrito ao sistema de arquivos de LFS. Isso acontece pois o shell Bash é informado que $LFS agora é o diretório raiz (/).

Perceba que /tools/bin não está no PATH. Isso significa que o conjunto de ferramentas cruzadas não mais será usado no ambiente chroot.

Note que o prompt de **bash** dirá I have no name! Isso é normal, pois 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 (reiniciar, por exemplo), então certifique-se que os sistemas de arquivos virtuais de kernel estejam montados como explicado em Seção 7.3.2, “Montando e Povoando /dev” e Seção 7.3.3, “Montando Sistemas de Arquivos Virtuais de Kernel” e entre no chroot novamente antes de continuar a instalação.

7.5. Criando Diretórios

É tempo de criar a estrutura completa no sistema de arquivos LFS.

Crie alguns diretórios de nível de raiz que não estão no conjunto limitado exigido nos capítulos anteriores emitindo o seguinte comando:

Nota

Alguns dos diretórios abaixo já foram criados anteriormente com instruções explícitas ou quando da instalação de alguns pacotes. Elas estão repetidas abaixo para completude.

```
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/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 para todos os diretórios. Nos comandos acima, duas mudanças são feitas—uma para o diretório home da(o) usuária(o) `root`, e outra para os diretórios para arquivos temporários.

A primeira mudança de modo assegura que nem qualquer pessoa possa entrar no diretório `/root`—o mesmo que uma(m) usuária(o) normal faria com o diretório home dela ou dele. A segunda mudança de modo garante que qualquer usuária(o) possa escrever nos diretórios `/tmp` e `/var/tmp`, mas não possa remover deles os arquivos de outras(os) usuárias(os). Essa última é proibida pelo assim chamado “sticky bit”, o bit mais alto (1) na máscara de bits 1777.

7.5.1. Nota de conformidade FHS

A árvore de diretório 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 alguns diretórios tais como `/usr/local/games` e `/usr/share/games`. Nós criamos apenas os diretórios que são necessários. Entretanto, sinta-se livre para criar esses diretórios.

7.6. Criando Arquivos Essenciais e Links Simbólicos

Historicamente, o Linux mantém uma lista dos sistemas de arquivos montados no arquivo `/etc/mtab`. Kernels modernos mantém essa lista internamente e expõem ela para a(o) usuária(o) via sistema de arquivos `/proc`. Para satisfazer utilitários que esperam a presença de `/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 a(o) usuária(o) `root` seja capaz de logar e para que o nome “root” seja reconhecido, precisa 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
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

A senha atual para `root` será configurada mais tarde.

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:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
input:x:24:
mail:x:34:
kvm:x:61:
uuidd:x:80:
wheel:x:97:
nogroup:x:99:
users:x:999:
EOF
```

Os grupos criados não são parte de qualquer padrão—eles são grupos decididos em parte pelas exigências da configuração de Udev no Capítulo 9, e em parte pelas convenções comuns empregadas por um número de distribuições Linux existentes. Em adição, algumas suítes de teste dependem de usuárias(os) ou grupos específicos. A Base Padrão Linux (Linux Standard Base - LSB, disponível em *http://refspecs.linuxfoundation.org/lsb.shtml*) apenas recomenda

que, além do grupo `root` com um ID de Grupo (GID) de 0, um grupo `bin` com um GID de 1 esteja presente. Todos os outros nomes de grupo e GIDs podem ser escolhidos livremente pela(o) administradora(r) de sistema uma vez que aplicativos bem escritos não dependem de números de GID, mas sim usam o nome do grupo.

Alguns testes em Capítulo 8 precisam de uma(m) usuária(o) regular. Nós adicionamos essa(e) usuária(o) 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` foram criados, 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 log para registrar informação tais como quem esteve logada(o) no sistema e quando. Entretanto, esses aplicativos não escreverão nos arquivos de log se eles já não existirem. Inicialize os arquivos de log 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) 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.

7.7. Libstdc++ oriundo de GCC-11.2.0, Passagem 2

Quando da construção de gcc-pass2 nós tivemos que adiar a instalação da biblioteca padrão C++, pois nenhum compilador adequado estava disponível para compilá-la. Nós não poderíamos usar o compilador construído naquela seção, por causa de que ele é um compilador nativo e não deveria ser usado do lado de fora do chroot e riscos de poluir as bibliotecas com alguns componentes do anfitrião.

Tempo aproximado de construção: 0,8 UPC
Espaço em disco exigido: 1,1 GB

7.7.1. Instalação de Libstdc++ Alvo

Nota

Libstdc++ é parte dos fontes de GCC. Você deveria primeiro desempacotar o tarball de GCC e mudar para o diretório `gcc-11.2.0`.

Crie um link que existe quando da construção de libstdc++ na árvore de gcc:

```
ln -s gthr-posix.h libgcc/gthr-default.h
```

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

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

Prepare libstdc++ para compilação:

```
../libstdc++-v3/configure           \
    CXXFLAGS="-g -O2 -D_GNU_SOURCE" \
    --prefix=/usr                   \
    --disable-multilib              \
    --disable-nls                   \
    --host=$(uname -m)-lfs-linux-gnu \
    --disable-libstdcxx-pch
```

O significado das opções de configure:

`CXXFLAGS="-g -O2 -D_GNU_SOURCE"`
Esses sinalizadores são passados pelo Makefile de nível de topo quando da feitura de uma construção completa de GCC.

`--host=$(uname -m)-lfs-linux-gnu`
Nós temos que imitar o que teria acontecido se esse pacote fosse construído como parte de uma construção completa de compilador. Essa chave teria sido passada para configure pelo maquinário de construção do GCC.

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

Compile libstdc++ executando:

```
make
```

Instale a biblioteca:

```
make install
```

Detalhes acerca deste pacote estão localizados em Seção 8.26.2, “Conteúdo de GCC.”

7.8. Gettext-0.21

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 da(o) usuária(o).

Tempo aproximado de construção: 1,6 UPC
Espaço em disco exigido: 280 MB

7.8.1. Instalação de Gettext

Para nosso conjunto temporário de ferramentas, nós apenas precisamos instalar três aplicativos originários de Gettext.

Prepare Gettext para compilação:

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

O significado da opção de configure:

`--disable-shared`
Nós não precisamos instalar quaisquer das bibliotecas compartilhadas de 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 em Seção 8.31.2, “Conteúdo do Gettext.”

7.9. Bison-3.8.2

O pacote Bison contém um gerador de analisador.

Tempo aproximado de construção: 0,3 UPC
Espaço em disco exigido: 50 MB

7.9.1. Instalação de Bison

Prepare Bison para compilação:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2
```

O significado da nova opção de configure:

`--docdir=/usr/share/doc/bison-3.8.2`
Isso diz ao sistema de construção para instalar documentação de bison em um diretório versionado.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados em Seção 8.32.2, “Conteúdo do Bison.”

7.10. Perl-5.34.0

O pacote Perl contém o Practical Extraction and Report Language.

Tempo aproximado de construção: 1,6 UPC
Espaço em disco exigido: 272 MB

7.10.1. Instalação de Perl

Prepare Perl para compilação:

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.34/core_perl      \
             -Darchlib=/usr/lib/perl5/5.34/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.34/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.34/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.34/vendor_perl \
             -Dvendorarch=/usr/lib/perl5/5.34/vendor_perl
```

O significado das novas opções de 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 não essencial.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados em Seção 8.41.2, “Conteúdo do Perl.”

7.11. Python-3.10.2

O pacote Python 3 contém o ambiente Python de desenvolvimento. Ele é útil para programação orientada a objetos, escrita de scripts, prototipagem de aplicativos grandes, ou desenvolvimento de aplicações inteiras.

Tempo aproximado de construção: 1,2 UPC
Espaço em disco exigido: 359 MB

7.11.1. Instalação de Python

Nota

Existem dois arquivos de pacotes cujos nomes se iniciam com “python”. Aquele a se extrair a partir dele é `Python-3.10.2.tar.xz` (perceba a primeira letra maiúscula).

Prepare Python para compilação:

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

O significado da opção de configure:

`--enable-shared`
Essa chave impede instalação de bibliotecas estáticas.

`--without-ensurepip`
Essa chave desabilita o instalador de pacote de Python, o qual não é necessário neste estágio.

Compile o pacote:

```
make
```

Nota

Alguns módulos de Python 3 não podem ser construídos agora, por causa de que as dependências não estão instaladas ainda. O sistema de construção ainda tenta construí-las, entretanto, de forma que a compilação de alguns arquivos falhará e a mensagem de compilador talvez pareça indicar “fatal error”. A mensagem deveria ser ignorada. Apenas tenha certeza de que o comando de nível de topo **make** não tenha falhado. Os módulos opcionais não são necessários agora e eles serão construídos em Capítulo 8.

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados em Seção 8.50.2, “Conteúdo do Python 3.”

7.12. Texinfo-6.8

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

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 109 MB

7.12.1. Instalação de Texinfo

Primeiro, conserte um problema ao construir o pacote com Glibc-2.34 ou posterior:

```
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c
```

Prepare Texinfo para compilação:

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

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados em Seção 8.67.2, “Conteúdo do Texinfo.”

7.13. Util-linux-2.37.4

O pacote Util-linux contém diversos aplicativos utilitários.

Tempo aproximado de construção: 0,7 UPC
Espaço em disco exigido: 129 MB

7.13.1. Instalação de Util-linux

O FHS recomenda usar o diretório `/var/lib/hwclock` em vez do usual diretório `/etc` como a localização para o arquivo `adjtime`. Crie esse diretório com:

```
mkdir -pv /var/lib/hwclock
```

Prepare Util-linux para compilação:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime    \
            --libdir=/usr/lib    \
            --docdir=/usr/share/doc/util-linux-2.37.4 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            runstatedir=/run
```

O significado das opções de configure:

`ADJTIME_PATH=/var/lib/hwclock/adjtime`
Isso configura a localização 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 outra localização, 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 de Python. Ela evita tentar construir ligações desnecessárias.

`runstatedir=/run`
Essa chave configura corretamente a localização do soquete usado por **uuidd** e `libuuid`.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Detalhes acerca deste pacote estão localizados em Seção 8.72.2, “Conteúdo do Util-linux.”

7.14. Limpando e Salvando o Sistema Temporário

7.14.1. 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, os arquivos .la de libtool somente são úteis quando vinculados com bibliotecas estáticas. Eles são desnecessários e potencialmente danosos quando do uso de bibliotecas compartilhadas dinâmicas, especialmente quando do uso de sistemas de construção não autotools. Enquanto ainda no chroot, remova aqueles arquivos agora:

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

O tamanho atual de 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
```

7.14.2. Cópia de segurança

Neste ponto os aplicativos e bibliotecas essenciais foram criados e seu sistema LFS atual está em um bom estado. Seu sistema pode agora 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 opção para recuperar. Infelizmente, todos os arquivos temporários serão removidos, também. Para evitar desperdiçar tempo extra para refazer tudo o que foi construído com sucesso, criar uma cópia de segurança do sistema LFS atual talvez se prove útil.

Nota

Todos os passos restantes nesta seção são opcionais. Apesar disso, tão logo você comece a instalar pacotes em Capítulo 8, os arquivos temporários serão sobrescritos. Assim, talvez 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, você tem de deixar o ambiente chroot primeiro antes de continuar. A razão para isso é para conseguir acesso a locais do sistema de arquivos do lado de fora do ambiente chroot para armazenar/ler o arquivamento de cópia de segurança o qual não deveria ser colocado dentro da hierarquia de `$LFS` por razões de segurança.

Se você decidiu fazer uma cópia de segurança, então deixe o ambiente chroot:

```
exit
```

Importante

Todas as instruções seguintes são executadas por `root` em seu sistema anfitrião. Tome cuidado extra acerca dos comandos que você vai executar, uma vez que erros aqui podem modificar seu sistema anfitrião. Esteja ciente de que a variável de ambiente `LFS` está configurada para usuária(o) `lfs` por padrão, mas talvez *não* esteja configurada para `root`.

Sempre que comandos forem ser executados por `root`, tenha certeza de que você configurou `LFS`.

Isso foi discutido em Seção 2.6, “Configurando a Variável $LFS”.

Antes de fazer uma cópia de segurança, desmonte os sistemas de arquivos virtuais:

```
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Tenha certeza de que tem pelo menos 1 GB de espaço de disco livre (os tarballs de fonte serão incluídos no arquivamento de cópia de segurança) no sistema de arquivos contendo diretório onde você criar o arquivamento de cópia de segurança.

Note que as instruções abaixo especificam o diretório home da(o) usuária(o) `root` do sistema anfitrião, o qual tipicamente é encontrado no sistema de arquivos raiz.

Substitua `$HOME` por um diretório da sua escolha se você não quiser ter a cópia de segurança armazenada no diretório home de `root`.

Crie o arquivamento de cópia de segurança executando o seguinte comando:

Nota

Por causa de que o arquivamento de cópia de segurança é comprimido, dura um tempo relativamente longo (mais de 10 minutos) mesmo em um sistema razoavelmente rápido.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-11.1.tar.xz .
```

Nota

Se continuar para o capítulo 8, então não se esqueça de entrar novamente no ambiente chroot conforme explanado na caixa “Importante” abaixo.

7.14.3. Restauro

No caso de alguns erros tiverem sido feitos 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 `$LFS`, eles são incluídos no arquivamento de cópia de segurança também, de forma que eles não precisam ser baixados novamente. Após verificar que `$LFS` está configurada adequadamente, restaure a cópia de segurança executando os seguintes comandos:

Atenção

Os seguintes comandos são extremamente perigosos. Se você executar **rm -rf ./*** como a(o) usuária(o) root e você não mudar para o diretório $LFS ou a variável de ambiente `LFS` não estiver configurada para a(o) usuária(o) root, então isso destruirá seu sistema anfitrião inteiro. VOCÊ ESTÁ AVISADA(O).

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-11.1.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 em Seção 7.3, “Preparando Sistemas de Arquivos Virtuais de Kernel” e entre novamente no ambiente chroot (veja 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 simples. 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 saber para que cada pacote é usado e porque você (ou o sistema) talvez precise dele.

Nós não recomendamos usar otimizações. Elas podem fazer com que um aplicativo execute ligeiramente mais rápido, mas elas também talvez causem dificuldades de compilação e problemas quando executar o aplicativo. Se um pacote se recusar a compilar quando usar otimização, então tente compilá-lo sem otimização e veja se isso conserta o problema. Mesmo se o pacote compilar quando usar otimização, existe o risco de que ele talvez tenha sido compilado incorretamente devido às complexas interações entre o código e ferramentas de construção. Note também que as opções `-march` e `-mtune` usando valores não especificados no livro não foram testadas. Isso talvez cause problemas com os pacotes do conjunto de ferramentas (Binutils, GCC e Glibc). Os pequenos ganhos potenciais alcançados usando otimizações de compilador frequentemente são superados pelos riscos. Construtoras(es) de primeira vez de LFS são encorajadas(os) a construir sem otimizações personalizadas. O sistema subsequente ainda executará muito rápido e será estável ao mesmo tempo.

Antes das instruções de instalação, cada página de instalação fornece informação acerca 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 bibliotecas (juntamente com breves descrições) que o pacote instala.

Nota

Os valores de UPC e espaço de disco exigido incluem dados de suíte de teste para todos os pacotes aplicáveis em Capítulo 8. Os valores de UPC foram calculados usando um núcleo sozinho de CPU (-j1) para todas as operações.

8.1.1. Acerca de bibliotecas

Em geral, as(os) editoras(es) de LFS desencorajam construir e instalar bibliotecas estáticas. O propósito original para a maioria das bibliotecas estáticas tem sido tornado obsoleto 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 todos os aplicativos que usam a biblioteca estática precisarão ser vinculados de novo à nova biblioteca. Como o uso de bibliotecas estáticas nem sempre é óbvio, os aplicativos relevantes (e os procedimentos necessários para fazer a vinculação) talvez nem mesmo sejam conhecidos.

Nos procedimentos neste capítulo, nós removemos ou desabilitamos a instalação da maioria das bibliotecas estáticas. Usualmente isso é feito passando-se uma opção `--disable-static` para **configure**. Em outros casos, meios alternativos são necessários. Em uns poucos casos, especialmente glibc e gcc, o uso de bibliotecas estáticas permanece essencial para o processo geral de construção de pacote.

Para uma discussão mais completa acerca de bibliotecas, veja-se a discussão *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 permite monitorar a instalação de arquivos tornando fácil remover e atualizar pacotes. Assim como os arquivos binários e bibliotecas, um gerenciador de pacote lidará com a instalação de arquivos de configuração. 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 talvez esteja entre essas técnicas ou talvez seja uma combinação de duas ou mais dessas técnicas. Esta seção menciona brevemente problemas que talvez surjam quando da atualização de pacotes.

Algumas razões porque nenhum gerenciador de pacote é mencionado em LFS ou 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. Incluir uma que satisfaça todas as audiências é difícil.

Existem algumas dicas escritas no tópico acerca de 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 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 kernel 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 borda bem definida entre kernel e espaço de usuária(o). Especificamente, os cabeçalhos de API de Linux não precisam ser (e não deveriam ser, veja-se o próximo item) atualizados juntamente com o kernel. Você precisará reiniciar seu sistema para usar o kernel atualizado.
- Se os cabeçalhos de API de Linux ou Glibc precisarem ser atualizados para uma versão mais nova, (por exemplo, de glibc-2.31 para glibc-2.32), então é mais seguro reconstruir LFS. Ainda que você *talvez* seja capaz de reconstruir todos os pacotes na ordem de dependência deles, nós não recomendamos isso.
- 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 com a finalidade de vincular à biblioteca mais nova. (Note 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 nome `libfoo.so.1`. Se você atualizar o pacote para uma versão mais nova foo-1.2.4 que instala uma biblioteca compartilhada com nome `libfoo.so.2`. Nesse caso, quaisquer pacotes que estiverem dinamicamente vinculados à `libfoo.so.1` precisam ser recompilados para vincular à `libfoo.so.2` com a finalidade de usar a nova versão de biblioteca. Você não deveria remover as bibliotecas anteriores a menos que todos os pacotes dependentes sejam recompilados.
- Se um pacote contendo uma biblioteca compartilhada for atualizado, e o nome da biblioteca não mudar, porém o número de versão do **arquivo** de biblioteca decrescer (por exemplo, o nome da biblioteca é mantido como `libfoo.so.1`, porém o nome do arquivo de biblioteca é modificado de `libfoo.so.1.25` para `libfoo.so.1.24`), então você deveria remover o arquivo de biblioteca originário da versão previamente instalada (`libfoo.so.1.25` no caso). Ou, uma execução de **ldconfig** (por você mesmo usando uma linha de comando, ou pela instalação de algum pacote) reconfigurará o link simbólico `libfoo.so.1` para apontar para o antigo

arquivo de biblioteca, pois ele aparenta ter uma versão “mais nova”, uma vez que seu número de versão é mais largo. Essa situação talvez aconteceu se você teve que desatualizar um pacote, ou o pacote muda repentinamente o esquema de versionamento de 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` após atualização, listará o que está usando as versões antigas daquelas bibliotecas (substitua *`libfoo`* com o nome da biblioteca):

```
grep -l  -e 'libfoo.*deleted' /proc/*/maps |
   tr -cd 0-9\\n | xargs -r ps u
```

  Se 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 reexecutar aquele comando para confirmar que nada ainda está usando as bibliotecas deletadas.

- Se um binário ou uma biblioteca compartilhada for sobrescrito, então os processos usando o código ou dados no binário ou biblioteca talvez quebrem. A maneira correta para atualizar um binário ou uma biblioteca compartilhada sem causar quebra ao processo é removê-lo primeiro, então instalar a versão nova na posição. O comando **install** fornecido por Coreutils já implementou isso e a maioria dos pacotes usa ele para instalar 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 Mozilla JS em BLFS) apenas sobrescreve o arquivo se ele existir e causa uma quebra, de forma que 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 do esquema em particular.

8.2.2.1. Está Tudo na Minha Cabeça!

Sim, isso é uma técnica de gerenciamento de pacote. Algumas pessoas não encontram a necessidade para um gerenciador de pacote, pois elas conhecem os pacotes intimamente e sabem quais arquivos estão instalados por cada pacote. Algumas(ns) usuárias(os) também não precisam de qualquer gerenciamento de pacote, pois elas(es) planejam reconstruir o sistema inteiro quando um pacote for mudado.

8.2.2.2. Instalação em Diretórios Separados

Esse é um gerenciamento de pacote simplista que não necessita de qualquer pacote extra para gerenciar as instalações. Cada pacote é instalado em um diretório separado. Por exemplo, o pacote foo-1.1 é instalado em `/usr/pkg/foo-1.1` e um link simbólico é feito de `/usr/pkg/foo` para `/usr/pkg/foo-1.1`. Quando da instalação de uma nova versão foo-1.2, ela é instalada em `/usr/pkg/foo-1.2` e o link simbólico anterior é substituído por um link simbólico para a nova versão.

Variáveis de ambiente tais como `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` e `CPPFLAGS` precisam ser expandidas para incluir `/usr/pkg/foo`. Para mais que uns poucos pacotes, esse esquema se torna ingerenciável.

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 similar ao esquema anterior. Mas, em vez de fazer o link simbólico, 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 pela(o) usuária(o) para automatizar a criação, muitos gerenciadores de pacote tem sido escritos usando essa abordagem. Alguns dos populares inclui Stow, Epkg, Graft, e Depot.

A instalação precisa ser falseada, 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, considere que você está instalando um pacote libfoo-1.1. As seguintes instruções talvez não instalem adequadamente o pacote:

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

A instalação funcionará, mas os pacotes dependentes talvez não se vinculem à libfoo conforme você esperaria. Se você compilar um pacote que vincula à libfoo, então você talvez note 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 usar a estratégia `DESTDIR` para falsear a instalação do pacote. 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 talvez ou precise instalar manualmente o pacote, ou você talvez ache que é mais fácil instalar alguns pacotes problemáticos em `/opt`.

8.2.2.4. Baseado em Marca Temporal

Nessa técnica, um arquivo é marcado temporalmente antes da instalação do pacote. Após a instalação, um simples uso do comando **find** com as opções apropriadas pode gerar um registro de todos os arquivos instalados após o arquivo de marca temporal ser criado. Um gerenciador de pacote escrito com 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 qualquer marca temporal outra que 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 estão sendo instalados em 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 **cp**, **install**, **mv** e rastreando as chamadas de sistema que modificam o sistema de arquivos. Para que essa abordagem funcione, todos os executáveis precisam ser dinamicamente vinculados sem o bit suid ou sgid. Pré-carregar a biblioteca talvez cause alguns efeitos colaterais indesejados durante a instalação. Portanto, aconselha-se que se realize alguns testes para garantir que o gerenciador de pacote não quebre nada e registre todos os arquivos adequados.

A segunda técnica é usar **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 no gerenciamento de pacote estilo Link Simbólico. Após a instalação, um arquivamento de pacote é criado usando os arquivos instalados. Esse arquivamento é então usado para instalar o pacote tanto na máquina local quanto pode até ser usado para instalar o pacote em outras máquinas.

Essa abordagem é 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*.

Criação de arquivos pacote que incluem informação de dependência é complexa e está além do escopo de LFS.

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 Slackware, veja *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 a dica em *https://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

8.2.3. Implantando 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 LFS para outro computador com a mesma arquitetura que a do sistema base é tão simples quanto usar **tar** na partição LFS que contém o diretório raiz (cerca de 250MB descomprimido para uma construção base LFS), copiando aquele arquivo via transferência de rede ou CD-ROM para o novo sistema e expandindo-o. A partir daquele ponto, uns poucos arquivos de configuração terão que ser mudados. Arquivos de configuração que talvez 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 kernel personalizado talvez seja necessário ser construído para o novo sistema dependendo das diferenças entre hardware de sistema e a configuração original do kernel.

Nota

Tem havido alguns relatos de problemas quando da cópia entre arquiteturas similares, porém não idênticas. Por exemplo, o conjunto de instrução para um sistema Intel não é idêntico com um processador AMD, e versões posteriores de alguns processadores talvez tenham 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-5.13

O pacote Man-pages contém mais que 2.200 páginas de manual.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 33 MB

8.3.1. Instalação de Man-pages

Instale Man-pages executando:

```
make prefix=/usr install
```

8.3.2. Conteúdo de Man-pages

Arquivos instalados: várias páginas de manual

Breves Descrições

`man pages` Descreve funções da linguagem de programação C, arquivos importantes de dispositivo e arquivos significantes de configuração

8.4. Iana-Etc-20220207

O pacote Iana-Etc fornece dados para serviços e protocolos de rede.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 4,7 MB

8.4.1. Instalação de Iana-Etc

Para esse pacote, nós apenas precisamos copiar os arquivos para o lugar:

```
cp services protocols /etc
```

8.4.2. Conteúdo de Iana-Etc

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

Breves Descrições

`/etc/protocols` Descreve os vários protocolos DARPA de Internet que estão disponíveis a partir do subsistema TCP/IP

`/etc/services` Fornece um mapeamento entre nomes textuais amigáveis para serviços de internet e seus números de porta atribuídos e tipos de protocolos não expostos

8.5. Glibc-2.35

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

Tempo aproximado de construção: 24 UPC
Espaço em disco exigido: 2,8 GB

8.5.1. Instalação de Glibc

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

```
patch -Np1 -i ../glibc-2.35-fhs-1.patch
```

A documentação de Glibc recomenda construir 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 Glibc para compilação:

```
../configure --prefix=/usr                            \
             --disable-werror                         \
             --enable-kernel=3.2                      \
             --enable-stack-protector=strong          \
             --with-headers=/usr/include              \
             libc_cv_slibdir=/usr/lib
```

O significado das opções de configure:

`--disable-werror`
Essa opção desabilita a opção -Werror passada para GCC. Isso é necessário para a execução da suíte de teste.

`--enable-kernel=3.2`
Essa opção diz ao sistema de construção que este glibc talvez seja usado com kernels tão antigos quanto 3.2. 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 de sistema adicionando código extra para verificar estouros de buffer, tais como ataques de esmagamento de pilha.

`--with-headers=/usr/include`
Essa opção diz ao sistema de construção onde encontrar os cabeçalhos de API de kernel.

libc_cv_slibdir=/usr/lib

Essa variável configura a biblioteca correta para todos os sistemas. Nós não queremos que lib64 seja usada.

Compile o pacote:

```
make
```

Importante

Nesta seção, a suíte de teste para Glibc é considerada crítica. Não pule sob qualquer circunstância.

Geralmente uns poucos testes não passam. As falhas de teste listadas abaixo são usualmente seguras ignorar.

```
make check
```

Você talvez veja algumas falhas de teste. A suíte de teste de Glibc é de alguma forma dependente do sistema anfitrião. Umas poucas falhas saídas de mais que 4.200 testes geralmente podem ignoradas. Esta é uma lista dos problemas mais comuns vistos para versões recentes de LFS:

- *io/tst-lchmod* é conhecido por falhar no ambiente chroot de LFS.
- *misc/tst-ttyname* é conhecido por falhar no ambiente chroot de LFS.
- O teste *nss/tst-nss-files-hosts-multi* é conhecido por falhar se o sistema não tiver endereços IP não loopback.

Mesmo sendo uma mensagem inofensiva, o estágio de instalação de Glibc reclamará acerca da ausência de `/etc/ld.so.conf`. Impeça esse alerta com:

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

Conserte o Makefile para pular uma verificação de sanidade desnecessária que falha no ambiente parcial de LFS:

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

Instale o pacote:

```
make install
```

Conserte caminho codificado rigidamente para o carregador de executável em script **ldd**:

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

Instale o arquivo de configuração e diretório de tempo de execução para **nscd**:

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

Em seguida, instale os locales que podem fazer o sistema responder em um idioma diferente. Nenhum dos locales é exigido, mas se algum deles estiver faltando, então as suítes de teste de futuros pacotes pulariam casos de teste importantes.

Locales individuais podem ser instalados usando o aplicativo **localedef**. Por exemplo, o segundo comando **localedef** abaixo combina a definição de locale independente de carácter `/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 locales necessário para a cobertura ótima de testes:

```
mkdir -pv /usr/lib/locale
localedef -i POSIX -f UTF-8 C.UTF-8 2> /dev/null || true
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f 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
```

Em adição, instale o locale para seu próprio país, idioma e conjunto de caracteres.

Alternativamente, instale todos os locales listados no arquivo `glibc-2.35/localedata/SUPPORTED` (inclui cada locale listado acima e muitos mais) de uma vez com o seguinte comando consumidor de tempo:

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

Então, use o comando **localedef** para criar e instalar locales não listados no arquivo `glibc-2.35/localedata/SUPPORTED` quando você precisar deles. Por exemplo, os seguintes dois locales são necessários para alguns testes posteriormente neste capítulo:

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

Nota

Glibc agora usa libidn2 quando da resolução de nomes internacionalizados de domínio. Essa é uma dependência de tempo de execução. Se essa capacidade for necessária, então as instruções para instalar libidn2 estão na *página libidn2 de BLFS*.

8.5.2. Configurando Glibc

8.5.2.1. Adicionando nsswitch.conf

O arquivo `/etc/nsswitch.conf` precisa ser criado, pois os padrões de Glibc não funcionam bem em um ambiente em rede.

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

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

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

8.5.2.2. Adicionando dados de fuso horário

Instale e configure os dados de fuso horário com o seguinte:

```
tar -xf ../../tzdata2021e.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 executando o seguinte script:

```
tzselect
```

Depois de responder à umas poucas perguntas sobre a localização, o script 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 script, mas podem ser usados.

Então crie o arquivo `/etc/localtime` executando:

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

Substitua *<xxx>* com o 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 `/lib` e `/usr/lib` por bibliotecas dinâmicas que são necessárias para aplicativos assim que são executados. Entretanto, se existirem bibliotecas em outros diretórios diferentes de `/lib` e `/usr/lib`, então 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"
# Begin /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 include são uma linha especificando o caminho de biblioteca desejado. Para adicionar essa capacidade, execute os seguintes comandos:

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

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

8.5.3. Conteúdo de Glibc

Aplicativos instalados: gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (link simbólico para ld-linux-x86-64.so.2 ou ld-linux.so.2), locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, e zic

Bibliotecas instaladas: ld-linux-x86-64.so.2, ld-linux.so.2, libBrokenLocale.{a,so}, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libc_malloc_debug.so, libcrypt.{a,so}, libdl.{a,so.2}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.so.1, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so.0}, libresolv.{a,so}, librt.{a,so.1}, libthread_db.so, e libutil.{a,so.1}

Diretórios instalados: /usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo, /var/cache/nscd, e /var/lib/nss_db

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 de **iconv**
ldconfig	Configura as ligações de tempo de execução do vinculador dinâmico
ldd	Reporta quis bibliotecas compartilhadas são exigidas por cada dado aplicativo ou biblioteca compartilhada
lddlibc4	Auxilia **ldd** com arquivos objeto. Isso não existe em arquiteturas mais novas como x86_64
locale	Imprime várias informações sobre o locale atual
localedef	Compila especificações de locale
makedb	Cria um banco de dados simples a partir de uma entrada textual
mtrace	Lê e interpreta um arquivo de rastreamento de memória e exibe um resumo em formato legível por humanos
nscd	Um daemon que fornece um cache para as solicitações de serviço de nomes mais comuns
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 usuário sobre a localização do sistema e reporta 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 ajudador para executáveis de bibliotecas compartilhadas
`libBrokenLocale`	Usado internamente por Glibc como um hack grosseiro para executar aplicativos quebrados (por exemplo, alguns aplicativos Motif). Veja comentários em `glibc-2.35/locale/broken_cur_max.c` para mais informação
`libanl`	Uma biblioteca assíncrona de pesquisa de nomes
`libc`	A biblioteca C principal
`libc_malloc_debug`	Liga verificação de alocação de memória quando pré-carregada
`libcrypt`	A biblioteca de criptografia
`libdl`	Biblioteca fictícia que não contém funções. Anteriormente era a biblioteca de interface do vinculador dinâmico, cujas funções agora estão em `libc`
`libg`	Biblioteca fictícia que não contém funções. Anteriormente era uma biblioteca de tempo de execução para **g++**
`libm`	A biblioteca matemática
`libmvec`	A biblioteca de vetor matemático, vinculada conforme necessária quando `libm` for usada

`libmcheck`	Liga verificação de alocação de memória quando quando vinculada para
`libmemusage`	Usado por **memusage** para ajudar a coletar informação sobre o uso de memória de um aplicativo
`libnsl`	A biblioteca de serviços de rede, agora obsoleta
`libnss_*`	Os módulos de Name Service Switch, contendo funções para resolução de nomes de hosts, nomes de usuárias(os), nomes de grupos, pseudônimos, serviços, protocolos, etc. Carregados por `libc` conforme a configuração em `/etc/nsswitch.conf`
`libpcprofile`	Pode ser pré-carregada para PC perfilar um executável
`libpthread`	Biblioteca fictícia que não contém funções. Anteriormente continha funções fornecendo a maior parte das interfaces especificadas pela Extensão POSIX.1b de Tempo Real, agora as funções estão em `libc`
`libresolv`	Contém funções para criação, envio e interpretação de pacotes para os servidores de nomes de domínio de Internet
`librt`	Contém funções fornecendo a maior parte das interfaces especificadas pela Extensão POSIX.1b de Tempo Real
`libthread_db`	Contém funções úteis para construir depuradores para aplicativos de múltiplas camadas
`libutil`	Biblioteca fictícia que não contém funções. Anteriormente continha código para funções “standard” usadas em muitos utilitários Unix. Essas funções agora estão em `libc`

8.6. Zlib-1.2.11

O pacote Zlib contém rotinas de compressão e descompressão usadas por alguns aplicativos.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 5,0 MB

8.6.1. Instalação de Zlib

Prepare Zlib para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

Remova uma biblioteca estática inútil:

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

8.6.2. Conteúdo de Zlib

Bibliotecas instaladas: libz.so

Descrições Curtas

`libz` Contém funções de compressão e descompressão usadas por alguns aplicativos

8.7. Bzip2-1.0.8

O pacote Bzip2 contém aplicativos para comprimir e descomprimir arquivos. Comprimir arquivos de texto com **bzip2** gera uma muito melhor percentagem de compressão que com o tradicional **gzip**.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 7,2 MB

8.7.1. Instalação de Bzip2

Aplique um patch 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 na localização correta:

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

Prepare Bzip2 para compilação com:

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

O significado do parâmetro de 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 de Bzip2 a 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 de **bzip2** com 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 de 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 bzmore), e bzmore
Bibliotecas instaladas: libbz2.so
Diretórios instalados: /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 **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

`libbz2` A biblioteca que implementa compressão de dados de classificação de blocos sem perdas, usando o algoritmo Burrows-Wheeler

8.8. Xz-5.2.5

O pacote Xz contém aplicativos para compressão e descompressão de arquivos. Ele fornece capacidades para os formatos de compressão lzma e o mais novo xz. Comprimir arquivos de texto com **xz** gera uma melhor percentagem de compressão que os tradicionais comandos **gzip** ou **bzip2**.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 15 MB

8.8.1. Instalação de Xz

Prepare Xz para compilação com:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.2.5
```

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.8.2. Conteúdo de Xz

Aplicativos instalados: lzcat (link para xz), lzcmp (link para xzdiff), lzdiff (link para xzdiff), lzegrep (link para xzgrep), lzfgrep (link para xzgrep), lzgrep (link para xzgrep), lzless (link para xzless), lzma (link para xz), lzmadec, lzmainfo, lzmore (link para xzmore), unlzma (link para xz), unxz (link para xz), xz, xzcat (link para xz), xzcmp (link para xzdiff), xzdec, xzdiff, xzegrep (link para xzgrep), xzfgrep (link para xzgrep), xzgrep, xzless, e xzmore
Bibliotecas instaladas: liblzma.so
Diretórios instalados: /usr/include/lzma e /usr/share/doc/xz-5.2.5

Descrições Curtas

lzcat Descomprime para a saída padrão
lzcmp Executa **cmp** em arquivos comprimidos LZMA
lzdiff Executa **diff** em arquivos comprimidos LZMA
lzegrep Executa **egrep** em arquivos comprimidos LZMA
lzfgrep Executa **fgrep** em arquivos comprimidos LZMA
lzgrep Executa **grep** em arquivos comprimidos LZMA
lzless Executa **less** em arquivos comprimidos LZMA
lzma Comprime ou descomprime arquivos usando o formato LZMA

lzmadec Um decodificador pequeno e rápido para arquivos comprimidos LZMA

lzmainfo Exibe informação armazenada no cabeçalho de arquivo comprimido com LZMA

lzmore Executa **more** em arquivos comprimidos LZMA

unlzma Descomprime arquivos usando o formato LZMA

unxz Descomprime arquivos usando o formato XZ

xz Comprime ou descomprime arquivos usando o formato XZ

xzcat Descomprime para a saída padrão

xzcmp Executa **cmp** em arquivos comprimidos XZ

xzdec Um decodificador pequeno e rápido para arquivos comprimidos XZ

xzdiff Executa **diff** em arquivos comprimidos XZ

xzegrep Executa **egrep** em arquivos comprimidos XZ

xzfgrep Executa **fgrep** em arquivos comprimidos XZ

xzgrep Executa **grep** em arquivos comprimidos XZ

xzless Executa **less** em arquivos comprimidos XZ

xzmore Executa **more** em arquivos comprimidos XZ

`liblzma` A biblioteca que implementa compressão de dados de classificação de blocos, sem perdas, usando o algoritmo de cadeia Lempel-Ziv-Markov

8.9. Zstd-1.5.2

Zstandard é um algoritmo de tempo real de compressão, 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: 1,1 UPC
Espaço em disco exigido: 55 MB

8.9.1. Instalação de Zstd

Compile o pacote:

```
make
```

Nota

Na saída de teste existem muitos lugares que indicam 'failed'. Essas são esperadas e apenas 'FAIL' é uma falha atual de teste. Não deveriam existir falhas de teste.

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make prefix=/usr install
```

Remova a biblioteca estática:

```
rm -v /usr/lib/libzstd.a
```

8.9.2. Conteúdo de Zstd

Aplicativos instalados: zstd, zstdcat (link para zstd), zstdgrep, zstdless, zstdmt (link para zstd), e unzstd (link para zstd)
Bibliotecas instaladas: 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

8.10. File-5.41

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

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 15 MB

8.10.1. Instalação de File

Prepare File para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.10.2. Conteúdo de File

Aplicativos instalados: file
Bibliotecas instaladas: 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, usado pelo aplicativo **file**

8.11. Readline-8.1.2

O pacote Readline é um conjunto de bibliotecas que oferecem edição de linha de comando e capacidades de histórico.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 15 MB

8.11.1. Instalação de Readline

Reinstalar Readline causará as bibliotecas antigas serem movidas para <libraryname>.old. Mesmo que isso normalmente não seja um problema, em alguns casos isso pode deflagrar um defeito de vinculação em **ldconfig**. Isso pode ser evitado executando os seguintes dois seds:

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

Prepare Readline para compilação:

```
./configure --prefix=/usr    \
            --disable-static \
            --with-curses    \
            --docdir=/usr/share/doc/readline-8.1.2
```

O significado da opção de configure:

`--with-curses`
Essa opção diz a Readline que ela pode encontrar as funções de biblioteca de termcap na biblioteca curses, em vez de uma biblioteca termcap separada. Ela permite a geração de um arquivo `readline.pc` correto.

Compile o pacote:

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

O significado da opção de make:

`SHLIB_LIBS="-lncursesw"`
Essa opção força Readline a vincular com 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-8.1.2
```

8.11.2. Conteúdo de Readline

Bibliotecas instaladas: libhistory.so e libreadline.so
Diretórios instalados: /usr/include/readline e /usr/share/doc/readline-8.1.2

Descrições Curtas

`libhistory`	Fornece uma consistente interface de usuária(o) para recordar linhas de histórico
`libreadline`	Fornece um conjunto de comandos para manipular texto digitado em uma sessão interativa de um aplicativo

8.12. M4-1.4.19

O pacote M4 contém um processador de macro.

Tempo aproximado de construção: 0,7 UPC
Espaço em disco exigido: 49 MB

8.12.1. Instalação de M4

Prepare M4 para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.12.2. 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 pela(o) usuária(o) 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

8.13. Bc-5.2.2

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: 7,1 MB

8.13.1. Instalação de Bc

Prepare Bc para compilação:

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

O significado das opções de configure:

`CC=gcc`
Esse parâmetro especifica o compilador a usar.

`-O3`
Especifica a optimização a usar.

`-G`
Omite partes da suíte de teste que não funcionariam sem um GNU bc presente.

Compile o pacote:

```
make
```

Para testar bc, execute:

```
make test
```

Instale o pacote:

```
make install
```

8.13.2. Conteúdo de Bc

Aplicativos instalados: bc e dc

Descrições Curtas

bc Uma calculadora de linha de comando

dc Uma calculadora de linha de comando de entrada polonesa - reversa

8.14. Flex-2.6.4

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

Tempo aproximado de construção: 0,4 UPC
Espaço em disco exigido: 32 MB

8.14.1. Instalação de 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), execute:

```
make check
```

Instale o pacote:

```
make install
```

Uns poucos aplicativos não sabem acerca de **flex** ainda e tentam executar seu predecessor, **lex**. Para suportar esses aplicativos, crie um link simbólico chamado `lex` que executa `flex` em modo de emulação **lex**:

```
ln -sv flex /usr/bin/lex
```

8.14.2. Conteúdo de Flex

Aplicativos instalados: flex, flex++ (link para flex), e lex (link para flex)
Bibliotecas instaladas: libfl.so
Diretórios instalados: /usr/share/doc/flex-2.6.4

Descrições Curtas

flex Uma ferramenta para gerar aplicativos que reconhecem padrões em texto; ele permite, para a versatilidade, especificar as regras para encontrar padrões, erradicando a necessidade de desenvolver um aplicativo especializado

flex++ Uma extensão de flex, é usada para gerar código e classes C++. É um link simbólico para **flex**

lex Um link simbólico que executa **flex** em modo de emulação **lex**

`libfl` A biblioteca `flex`

8.15. Tcl-8.6.12

O pacote Tcl contém a Tool Command Language, uma linguagem de script robusta de propósito geral. O pacote Expect é escrito na linguagem Tcl.

Tempo aproximado de construção: 3,4 UPC
Espaço em disco exigido: 87 MB

8.15.1. Instalação de Tcl

Esse pacote e os próximos dois (Expect e DejaGNU) são instalados para suportar a execução das suítes de teste para binutils e GCC e outros pacotes. Instalar três pacotes para propósitos de teste talvez pareça excessivo, mas é muito assegurador, se não essencial, saber que as ferramentas mais importantes estão funcionando adequadamente.

Primeiro, desempacote a documentação executando o seguinte comando:

```
tar -xf ../tcl8.6.12-html.tar.gz --strip-components=1
```

Prepare Tcl para compilação:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            $([ "$(uname -m)" = x86_64 ] && echo --enable-64bit)
```

O significado das opções de configure:

`$([ "$(uname -m)" = x86_64 ] && echo --enable-64bit)`
A construção *`$(<shell command>)`* é substituída pela saída do comando de shell. Aqui essa saída é vazia se executada em uma máquina de 32 bits, e é *`--enable-64bit`* se executada em uma máquina de 64 bits.

Construa o pacote:

```
make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|"  \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.3|/usr/lib/tdbc1.1.3|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3/library|/usr/lib/tcl8.6|" \
    -e "s|$SRCDIR/pkgs/tdbc1.1.3|/usr/include|"            \
    -i pkgs/tdbc1.1.3/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.2.2|/usr/lib/itcl4.2.2|" \
    -e "s|$SRCDIR/pkgs/itcl4.2.2/generic|/usr/include|"    \
    -e "s|$SRCDIR/pkgs/itcl4.2.2|/usr/include|"            \
    -i pkgs/itcl4.2.2/itclConfig.sh

unset SRCDIR
```

As várias instruções “sed” após o comando “make” removem referências ao diretório de construção dos arquivos de configuração e as substituem com o diretório de instalação. Isso não é obrigatório para o restante de LFS, porém talvez seja necessário caso um pacote construído posteriormente use Tcl.

Para testar os resultados, execute:

```
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 do Tcl. O próximo pacote, Expect, exige elas.

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

Agora faça um necessário link simbólico:

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Renomeie uma página de manual que conflita com uma página de manual de Perl:

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

Se você transferiu a documentação opcional, então instale ela executando os seguintes comandos:

```
mkdir -v -p /usr/share/doc/tcl-8.6.12
cp -v -r  ../html/* /usr/share/doc/tcl-8.6.12
```

8.15.2. Conteúdo de Tcl

Aplicativos instalados: tclsh (link to tclsh8.6) e tclsh8.6
Bibliotecas instaladas: libtcl8.6.so e libtclstub8.6.a

Descrições Curtas

tclsh8.6	O shell de comando de Tcl
tclsh	Um link para tclsh8.6
`libtcl8.6.so`	A biblioteca Tcl
`libtclstub8.6.a`	A biblioteca Stub de Tcl

8.16. Expect-5.45.4

O pacote Expect contém ferramentas para automatizar, via diálogos com script, aplicativos interativos tais como **telnet**, **ftp**, **passwd**, **fsck**, **rlogin**, e **tip**. Expect também é útil para testar esses mesmos aplicativos bem como facilitar todos os tipos de tarefas que são proibitivamente difíceis com qualquer outra coisa. A estrutura subjacente de DejaGnu é escrita em Expect.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 3,9 MB

8.16.1. Instalação de Expect

Prepare Expect para compilação:

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

O significado das opções de configure:

`--with-tcl=/usr/lib`
Esse parâmetro é necessário para dizer a **configure** onde o script **tclConfig.sh** está localizado.

`--with-tclinclude=/usr/include`
Isso explicitamente diz a Expect onde encontrar os cabeçalhos internos de Tcl.

Construa o pacote:

```
make
```

Para testar os resultados, execute:

```
make test
```

Instale o pacote:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.16.2. Conteúdo de Expect

Aplicativo instalado: expect
Biblioteca instalada: libexpect5.45.4.so

Descrições Curtas

expect Comunica-se com outros aplicativos interativos de acordo com um script

`libexpect-5.45.4.so` Contém funções que permitem a Expect ser usado como uma extensão Tcl ou ser usado diretamente a partir de C ou C++ (sem Tcl)

8.17. 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 Tcl (Tool Command Language).

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 6,9 MB

8.17.1. Instalação de DejaGNU

A(O) desenvolvedora(r) recomenda construir DejaGNU em um diretório dedicado à construção:

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

Prepare DejaGNU para compilação:

```
../configure --prefix=/usr
makeinfo --html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo --plaintext       -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Construa e instale o pacote:

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

Para testar os resultados, execute:

```
make check
```

8.17.2. Conteúdo de DejaGNU

Aplicativos instalados: dejagnu e runtest

Descrições Curtas

dejagnu Iniciador de comando auxiliar DejaGNU

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

8.18. Binutils-2.38

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

Tempo aproximado de construção: 6,1 UPC
Espaço em disco exigido: 4,6 GB

8.18.1. Instalação de Binutils

Verifique se os Pseudo Terminais (PTYs) estão funcionando adequadamente dentro do ambiente chroot executando um teste simples:

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

Esse comando deveria retornar o seguinte:

```
spawn ls
```

Se, ao invés, a saída incluir a mensagem abaixo, então o ambiente não está configurado para operação adequada de PTY. Esse problema precisa ser resolvido antes de executar as suítes de teste para Binutils e GCC:

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

A(O) desenvolvedora(r) fez um conjunto curto de comandos para fixar um problema quando do uso de binutils para construir alguns pacotes BLFS com Link Time Optimization (LTO) habilitado. Aplique ele agora:

```
patch -Np1 -i ../binutils-2.38-lto_fix-1.patch
```

Agora, faça um conserto identificado pela(o) desenvolvedora(r) que afeta a construção alguns pacotes:

```
sed -e '/R_386_TLS_LE /i \   || (TYPE) == R_386_TLS_IE \\' \
    -i ./bfd/elfxx-x86.h
```

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

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

Prepare Binutils para compilação:

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

O significado dos parâmetros de configure:

`--enable-gold`
Constrói o vinculador gold e instala ele como ld.gold (juntamente com o vinculador padrão).

--enable-ld=default
Constrói o vinculador bfd original e instala ele como ambos ld (o vinculador padrão) e ld.bfd.

--enable-plugins
Habilita suporte de plugin para o vinculador.

--enable-64-bit-bfd
Habilita suporte de 64 bits (em anfitriões com tamanhos de palavra mais estreitos). Talvez não seja necessário em sistemas de 64 bits, porém não causa dano.

--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 de make:

tooldir=/usr
Normalmente, o tooldir (o diretório onde os executáveis estarão ultimamente 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 pule sob quaisquer circunstâncias.

Teste os resultados:

```
make -k check
```

Instale o pacote:

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

Remova bibliotecas estáticas inúteis:

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

8.18.2. Conteúdo de Binutils

Aplicativos instalados:	addr2line, ar, as, c++filt, dwp, elfedit, gprof, ld, ld.bfd, ld.gold, nm, objcopy, objdump, ranlib, readelf, size, strings, e strip
Bibliotecas instaladas:	libbfd.so, libctf.so, libctf-nobfd.so, e libopcodes.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 ao endereço

ar	Cria, modifica e extrai a partir de arquivamentos
as	Um montador que monta a saída de **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
ld	Um vinculador que combina um número de objetos e arquivos de arquivamento em um arquivo, realocando seus dados e vinculando referências de símbolos
ld.gold	Uma versão reduzida de ld que suporta apenas o formato de arquivo de objeto elf
ld.bfd	Hard link para **ld**
nm	Lista os símbolos que ocorrem em um dado arquivo de objeto
objcopy	Traduz um tipo de arquivo de objeto em outro
objdump	Exibe informação sobre o dado arquivo de objeto, com opções controlando a informação particular a ser exibida; a informação mostrada é útil para programadores 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 de objeto realocáveis
readelf	Exibe informação sobre binários de tipo ELF
size	Lista os tamanhos de seção e o tamanho total para os arquivos de objeto dados
strings	Exibe, para cada arquivo dado, as sequências de caracteres imprimíveis que são de, no mínimo, o tamanho especificado (padronizado para quatro); para arquivos de objeto, ele imprime, por padrão, apenas 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 de arquivos de objeto
`libbfd`	A biblioteca de Descritor de Arquivo Binário
`libctf`	A biblioteca de suporte de depuração Compat ANSI-C Type Format
`libctf-nobfd`	Uma variante de libctf que não usa funcionalidade de libbfd
`libopcodes`	Uma biblioteca para lidar com opcodes—as versões de “texto legível” de instruções para o processador; é usado para construir utilitários como **objdump**

8.19. GMP-6.2.1

O pacote GMP contém bibliotecas matemáticas. Essas tem funções úteis para aritmética de precisão arbitrária.

Tempo aproximado de construção: 1,0 UPC
Espaço em disco exigido: 52 MB

8.19.1. Instalação de GMP

Nota

Se você estiver construindo para x86 de 32 bits, mas tem uma CPU 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 de configure abaixo com

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

Nota

As configurações padrão de 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 executando o seguinte:

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

Prepare GMP para compilação:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.2.1
```

O significado das novas opções de configure:

`--enable-cxx`
Esse parâmetro habilita suporte a C++

`--docdir=/usr/share/doc/gmp-6.2.1`
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 GMP nesta seção é considerada crítica. Não 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 é construído. Ocasionalmente, o código que detecta o processador identifica errado as capacidades de sistema e existirão erros nos testes ou outros aplicativos que usam as bibliotecas de gmp com a mensagem "Illegal instruction". Nesse caso, gmp deveria ser reconfigurado com a opção --build=x86_64-pc-linux-gnu e reconstruído.

Certifique-se de que todos os 197 testes na suíte de teste passaram. Verifique os resultados executando 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
```

8.19.2. Conteúdo de GMP

Bibliotecas instaladas: libgmp.so e libgmpxx.so
Diretório instalado: /usr/share/doc/gmp-6.2.1

Descrições Curtas

`libgmp` Contém funções matemáticas de precisão

`libgmpxx` Contém funções matemáticas de precisão C++

8.20. MPFR-4.1.0

O pacote MPFR contém funções para matemática de precisão múltipla.

Tempo aproximado de construção: 0,8 UPC
Espaço em disco exigido: 38 MB

8.20.1. Instalação de MPFR

Prepare MPFR para compilação:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.1.0
```

Compile o pacote e gere a documentação HTML:

```
make
make html
```

Importante

A suíte de teste para MPFR nesta seção é considerada crítica. Não pule sob quaisquer circunstâncias.

Teste os resultados e certifique-se de que todos os testes passaram:

```
make check
```

Instale o pacote e a documentação dele:

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

8.20.2. Conteúdo de MPFR

Biblioteca instalada: libmpfr.so
Diretório instalado: /usr/share/doc/mpfr-4.1.0

Descrições Curtas

`libmpfr` Contém funções matemáticas de precisão múltipla

8.21. MPC-1.2.1

O pacote MPC contém uma biblioteca para a aritmética de números complexos com precisão arbitrariamente alta e arredondamento correto de resultado.

Tempo aproximado de construção: 0,3 UPC
Espaço em disco exigido: 21 MB

8.21.1. Instalação de MPC

Prepare MPC para compilação:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.2.1
```

Compile o pacote e gere a documentação HTML:

```
make
make html
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote e a documentação dele:

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

8.21.2. Conteúdo de MPC

Biblioteca instalada: libmpc.so
Diretório instalado: /usr/share/doc/mpc-1.2.1

Descrições Curtas

`libmpc` Contém funções matemáticas complexas

8.22. Attr-2.5.1

O pacote attr contém utilitários para administrar os atributos estendidos sobre objetos de sistema de arquivos.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 4,1 MB

8.22.1. Instalação de Attr

Prepare Attr para compilação:

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.1
```

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, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.22.2. Conteúdo de Attr

Aplicativos instalados: attr, getfattr, e setfattr
Biblioteca instalada: libattr.so
Diretórios instalados: /usr/include/attr e /usr/share/doc/attr-2.5.1

Descrições Curtas

attr Estende atributos sobre objetos de sistemas de arquivos
getfattr Obtém os atributos estendidos de objetos de sistemas de arquivos
setfattr Configura os atributos estendidos de objetos de sistemas de arquivos
`libattr` Contém as funções de biblioteca para manipular atributos estendidos

8.23. Acl-2.3.1

O pacote Acl contém utilitários para administrar Listas de Controle de Acesso, as quais são usadas para definir direitos de acesso discricionários mais refinados para arquivos e diretórios.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 6,1 MB

8.23.1. Instalação de Acl

Prepare Acl para compilação:

```
./configure --prefix=/usr         \
            --disable-static      \
            --docdir=/usr/share/doc/acl-2.3.1
```

Compile o pacote:

```
make
```

Os testes de Acl precisam ser executados sobre um sistema de arquivos que suporte controles de acesso, após Coreutils ter sido construído com as bibliotecas de Acl. Se desejado, retorne a esse pacote e execute **make check** após Coreutils ter sido construído posteriormente neste capítulo.

Instale o pacote:

```
make install
```

8.23.2. Conteúdo de Acl

Aplicativos instalados: chacl, getfacl, e setfacl
Biblioteca instalada: libacl.so
Diretórios instalados: /usr/include/acl e /usr/share/doc/acl-2.3.1

Descrições Curtas

chacl Muda a lista de controle de acesso de um arquivo ou diretório

getfacl Obtém listas de controle de acesso de arquivo

setfacl Configura listas de controle de acesso de arquivo

`libacl` Contém as funções de biblioteca para manipular Listas de Controle de Acesso

8.24. Libcap-2.63

O pacote Libcap implementa as interfaces de espaço de usuária(o) para as capacidades POSIX 1003.1e disponíveis em kernels Linux. Essas capacidades são um particionamento de todo o poderoso privilégio de root em um conjunto de privilégios distintos.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 2,7 MB

8.24.1. Instalação de 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 de 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, execute:

```
make test
```

Instale o pacote:

```
make prefix=/usr lib=lib install
```

8.24.2. Conteúdo de Libcap

Aplicativos instalados: capsh, getcap, getpcaps, e setcap
Bibliotecas instaladas: libcap.so e libpsx.so

Descrições Curtas

capsh Um encapsulador de shell para explorar e restringir suporte de capacidade
getcap Examina capacidades de arquivo
getpcaps Exibe as capacidades sobre o(s) processo(s) consultado(s)
setcap Configura capacidades de arquivo
`libcap` Contém as funções de biblioteca para manipular capacidades POSIX 1003.1e
`libpsx` Contém funções para suportar semântica POSIX para chamadas de sistema associadas com a biblioteca pthread

8.25. Shadow-4.11.1

O pacote Shadow contém aplicativos para manipular senhas de uma maneira segura.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 49 MB

8.25.1. Instalação de Shadow

Nota

Se você gostaria de reforçar o uso de senhas fortes, então recorra a *https://www.linuxfromscratch.org/blfs/view/11.1/postlfs/cracklib.html* para instalar CrackLib antes de construir Shadow. Então adicione `--with-libcrack` ao comando **configure** abaixo.

Desabilite a instalação do aplicativo **groups** e suas páginas de manual, uma vez que Coreutils fornece uma versão melhor. Também, impeça a instalação de páginas de manual que já foram instaladas em Seção 8.3, “Man-pages-5.13”:

```
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 mais seguro *SHA-512* de encriptação de senha, o qual também permite senhas maiores que oito (08) caracteres. É também necessário mudar a localização obsoleta `/var/spool/mail` para caixas de correio de usuária(o) que Shadow usa por padrão pela localização `/var/mail` usada atualmente. E, livre-se de `/bin` e `/sbin` a partir de `PATH`, uma vez que eles são simples links simbólicos para seus homônimos em `/usr`.

Nota

Se `/bin` e (ou) `/sbin` forem preferidos para serem deixados em `PATH` por alguma razão, então modifique `PATH` em `.bashrc` após LFS ser construído.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD SHA512:' \
    -e 's:/var/spool/mail:/var/mail:'                 \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'                \
    -i etc/login.defs
```

Nota

Se você escolher construir Shadow com suporte CrackLib, então execute o seguinte:

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

Prepare Shadow para compilação:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc \
            --disable-static  \
            --with-group-name-max-length=32
```

O significado da opção de configure:

touch /usr/bin/passwd
O arquivo `/usr/bin/passwd` precisa existir, pois a localização dele é codificada rigidamente em alguns aplicativos, e se ele não existir, então a localização padrão não é correta.

`--with-group-name-max-length=32`
O nome de usuária(o) máximo é trinta e dois (32) caracteres. Torne o nome de grupo máximo o mesmo.

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
```

8.25.2. Configurando Shadow

Esse pacote contém utilitários para adicionar, modificar, e deletar usuárias(os) e grupos; configura e modifica suas senhas; e realiza outras tarefas administrativas. Para uma explanação completa do que *password shadowing* significa, veja o arquivo `doc/HOWTO` dentro da árvore desempacotada de fonte. Se usar suporte Shadow, então tenha na mente que aplicativos que necessitem verificar senhas (gerenciadores de tela, aplicativos de FTP, daemons pop3, etc.) precisam ser conformes com Shadow. Isto é, eles precisam ser capazes de trabalhar com senhas ocultas.

Para habilitar senhas ocultas, execute o seguinte comando:

```
pwconv
```

Para habilitar senhas ocultas de grupo, execute:

```
grpconv
```

A configuração padrão de Shadow para o utilitário **useradd** tem umas poucas ressalvas que precisam de alguma explanação. Primeiro, a ação padrão para o utilitário **useradd** é a de criar a(o) usuária(o) e um grupo de mesmo nome que a(o) usuária(o). Por padrão os números de ID de usuária(o) (UID) e ID de grupo (GID) iniciarão com 1000. Isso significa que se você não passar parâmetros para **useradd**, então cada usuária(o) será uma(m) membro de um grupo único no sistema. Se esse comportamento for indesejável, então você precisará passar um parâmetro de *`-g`* ou *`-N`* para **useradd** ou 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 atender às suas necessidades particulares. Crie ele com:

```
mkdir -p /etc/default
useradd -D --gid 999
```

Explanações de Parâmetro de `/etc/default/useradd`

`GROUP=999`
Esse parâmetro configura o início dos números de grupo usado no arquivo `/etc/group`. O valor particular 999 vem do parâmetro *`--gid`* acima. Você pode modificá-lo para qualquer coisa que deseje. Note que **useradd** nunca

reusará um UID ou GID. Se o número identificado nesse parâmetro for usado, então ele usará o próximo número disponível. Note também que se você não tiver um grupo com um ID igual a esse número em seu sistema na primeira vez que você usar **useradd** sem o parâmetro *-g*, então você receberá uma mensagem exibida no terminal que diz: `useradd: unknown GID 999`, apesar de a conta estar criada corretamente. Esse é o motivo pelo qual nós criamos o grupo `users` com esse ID de grupo em Seção 7.6, “Criando Arquivos Essenciais e Links Simbólicos”.

`CREATE_MAIL_SPOOL=yes`

Esse parâmetro faz com que **useradd** crie um arquivo de caixa de correio para a(o) usuária(o) recém criada(o). **useradd** tornará a propriedade de grupo desse arquivo para o grupo `mail` com permissões 0660. Se você preferisse que esses arquivos de caixa de correio não fossem criados por **useradd**, então execute o seguinte comando:

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

8.25.3. Configurando a senha de root

Escolha uma senha para a(o) usuária(o) *root* e configure ela executando:

```
passwd root
```

8.25.4. Conteúdo de Shadow

Aplicativos instalados: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, 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

Biblioteca instalada: libsubid.so

Diretório instalado: /etc/default

Descrições Curtas

chage Usado para alterar o número de dias máximo entre mudanças obrigatórias de senha

chfn Usado para alterar um nome completo da(o) usuária(o) e outra informação

chgpasswd Usado para atualizar senhas de grupo em modo de lote

chpasswd Usado para atualizar senhas de usuárias(os) em modo de lote

chsh Usado para alterar um shell de login padrão da(o) usuária(o)

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 uma(m) usuária(o)

gpasswd É usado para adicionar e deletar membros e administradoras(es) para grupos

groupadd Cria um grupo com o nome dado

groupdel Deleta o grupo com o nome dado

groupmems Permite que uma(m) usuária(o) administre sua própria lista de membros de grupo sem a exigência de privilégios de superusuária(o)

groupmod É usado para modificar o nome ou GID do grupo dado

grpck Verifica a integridade dos arquivos de grupo `/etc/group` e `/etc/gshadow`

grpconv Cria ou atualiza o arquivo de grupo de sombra a partir do arquivo de grupo normal

grpunconv Atualiza `/etc/group` a partir de `/etc/gshadow` e então deleta o último

lastlog Reporta o login mais recente de todas(os) as(os) usuárias(os) ou de uma(m) usuária(o) dada(o)

login É usado pelo sistema para permitir usuárias(os) logar

logoutd É um daemon 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 modificar 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 que uma conta não está disponível; projetado para ser usado como o shell padrão para contas que foram desabilitadas

passwd É usado para modificar a senha para uma conta de usuária(o) ou grupo

pwck Verifica a integridade dos arquivos de senha `/etc/passwd` e `/etc/shadow`

pwconv Cria ou atualiza o arquivo de senha de sombra a partir do arquivo de senha normal

pwunconv Atualiza `/etc/passwd` a partir de `/etc/shadow` e então deleta o último

sg Executa um comando dado enquanto o GID da(o) usuária(o) está configurado para aquele do grupo dado

su Executa um shell com IDs de usuária(o) e grupo substitutos

useradd Cria uma(m) usuária(o) nova(o) com o nome dado, ou atualiza a informação padrão de nova(o) usuária(o)

userdel Deleta a conta de usuária(o) dada

usermod É usado para modificar o nome de login da(o) usuária(o) dada(o), Identificação de Usuária(o) (UID), shell, grupo inicial, diretório home, etc.

vigr Edita os arquivos `/etc/group` ou `/etc/gshadow`

vipw Edita os arquivos `/etc/passwd` ou `/etc/shadow`

`libsubid` Biblioteca para processar intervalos subordinados de id para usuárias(os)

8.26. GCC-11.2.0

O pacote GCC contém a GNU compiler collection, o qual inclui os compiladores C e C++.

Tempo aproximado de construção: 153 UPC (com os testes)
Espaço em disco exigido: 4,3 GB

8.26.1. Instalação de GCC

Primeiramente, conserte um problema que quebra `libasan.a` quando da construção desse pacote com Glibc-2.34 ou posterior:

```
sed -e '/static.*SIGSTKSZ/d' \
    -e 's/return kAltStackSize/return SIGSTKSZ * 4/' \
    -i libsanitizer/sanitizer_common/sanitizer_posix_libcdep.cpp
```

Se construir em x86_64, então mude o nome de diretório padrão 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 de GCC recomenda construir GCC em um diretório de construção dedicado:

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

Prepare GCC para compilação:

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

Note que para outras linguagens de programação, existem alguns pré-requisitos que ainda não estão disponíveis. Veja-se a *página de GCC do Livro BLFS* para instruções sobre como construir todas as linguagens suportadas do GCC.

O significado dos novos parâmetros de configure:

`LD=ld`
Esse parâmetro induz o script configure a usar o ld instalado pelo binutils construído anteriormente neste capítulo, em vez da versão construída cruzadamente a qual de outra maneira seria usada.

`--with-system-zlib`
Essa chave diz a GCC para vincular à cópia instalada de sistema da biblioteca zlib, em vez de sua própria cópia interna.

Compile o pacote:

```
make
```

Importante

Nesta seção, a suíte de teste para GCC é considerada importante, porém ela toma um tempo longo. Construtoras(es) de primeira vez são encorajadas(os) a não pular ela. O tempo para executar os testes pode ser reduzido significantemente adicionando-se -jx ao comando make abaixo, onde x é o número de núcleos em seu sistema.

Um conjunto de testes na suíte de teste de GCC é conhecida por esgotar a pilha padrão, então aumente o tamanho de pilha antes de executar os testes:

```
ulimit -s 32768
```

Teste os resultados como uma(m) usuária(o) não privilegiada(o), porém não pare aos erros:

```
chown -Rv tester .
su tester -c "PATH=$PATH make -k check"
```

Para receber um sumário dos resultados de suíte de teste, execute:

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

Para apenas os sumários, entube a saída por grep -A7 Summ.

Resultados podem ser comparados com aqueles localizados em *https://www.linuxfromscratch.org/lfs/build-logs/11.1/* e *https://gcc.gnu.org/ml/gcc-testresults/*.

Oito testes relacionados ao analisador são conhecidos por falhar.

Um teste chamado `asan_test.C` é conhecido por falhar.

Em libstdc++, um teste chamado `49745.cc` é conhecido por falhar, pois as dependências de cabeçalho em glibc mudaram.

Em libstdc++, um teste de numeração de pontuação e seis testes relacionados a get_time são conhecidos por falhar. Essas são todas por causa das definições de locale em glibc que mudaram, porém libstdc++ atualmente não suporta essas mudanças.

Umas poucas falhas inesperadas não podem ser evitadas sempre. As(Os) desenvolvedoras(es) de GCC geralmente estão cientes desses problemas, mas ainda não os resolveram. A menos que os resultados de teste sejam amplamente diferentes daqueles na URL acima, é seguro continuar.

Instale o pacote e remova um diretório desnecessário:

```
make install
rm -rf /usr/lib/gcc/$(gcc -dumpmachine)/11.2.0/include-fixed/bits/
```

O diretório de construção de GCC é de propriedade de `tester` agora e a propriedade do diretório de cabeçalho instalado (e o conteúdo dele) estarão incorretos. Mude a propriedade para usuária(o) e grupo `root`:

```
chown -v -R root:root \
    /usr/lib/gcc/*linux-gnu/11.2.0/include{,-fixed}
```

Crie um link simbólico exigido por *FHS* por razões "históricas".

```
ln -svr /usr/bin/cpp /usr/lib
```

Adicione um link simbólico de compatibilidade para habilitar a construção de aplicativos com Link Time Optimization (LTO):

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/11.2.0/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'
```

Deveriam não existir erros, e a saída do último comando será (permitindo diferenças específicas de plataforma no nome de 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 iniciação corretos:

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

A saída do último comando deveria ser:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/../../../../lib/crt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/../../../../lib/crtn.o succeeded
```

Dependendo da arquitetura de sua máquina, o acima talvez difira levemente. A diferença será o nome do diretório depois de `/usr/lib/gcc`. A coisa importante a se olhar aqui é que **gcc** encontrou todos os três arquivos `crt*.o` sob o diretório `/usr/lib`.

Verifique que o compilador está procurando pelos arquivos de cabeçalho corretos:

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

Esse comando deveria retornar a seguinte saída:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/11.2.0/include-fixed
 /usr/include
```

Novamente, o diretório nomeado após seu triplet alvo talvez seja diferente do que o acima, dependendo da arquitetura de seu sistema.

Em seguida, verifique que o novo vinculador está sendo usado com os caminhos de procura corretos:

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

Referências a caminhos que tem componentes com '-linux-gnu' deveriam ser ignoradas, porém, do contrário, a saída 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 talvez veja uns poucos diretórios diferentes. Por exemplo, aqui está a saída 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 do último comando deveria ser:

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

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

```
grep found dummy.log
```

A saída do último comando deveria ser (permitindo diferenças específicas de plataforma no nome de vinculador dinâmico):

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

Se a saída não aparecer como mostrado acima ou não for recebida de jeito nenhum, então alguma coisa está seriamente errada. Investigue e retrace os passos para encontrar onde está o problema e corrija o mesmo. Quaisquer problemas precisão ser resolvidos antes de continuar com o processo.

Uma vez que tudo esteja funcionando corretamente, limpe 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
```

8.26.2. Conteúdo de 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}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++fs.a, libsupc++.a, libtsan.{a,so}, e libubsan.{a,so}

Diretórios instalados: /usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, e /usr/share/gcc-11.2.0

Descrições Curtas

c++ O compilador C++

cc O compilador C

cpp O preprocessador C; é usado pelo compilador para expandir as declarações #include, #define e similares nos arquivos fonte

g++ O compilador C++

gcc O compilador C

gcc-ar Um encapsulador em torno de **ar** que adiciona um plugin à linha de comando. Esse aplicativo é usado apenas para adicionar "link time optimization" e não é útil com as opções de construção padrão

gcc-nm Um encapsulador em torno de **nm** que adiciona um plugin à linha de comando. Esse aplicativo é usado apenas para adicionar "link time optimization" e não é útil com as opções de construção padrão

gcc-ranlib Um encapsulador em torno de **ranlib** que adiciona um plugin à linha de comando. Esse aplicativo é usado apenas para adicionar "link time optimization" e não é útil com as opções de construção padrão

gcov Uma ferramenta de teste de cobertura; usada para analisar aplicativos para determinar onde as otimizações terão mais 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 por GCC com LTO habilitado

`libasan` A biblioteca de tempo de execução do Address Sanitizer

`libatomic` Biblioteca de tempo de execução atômica interna do GCC

`libcc1` A biblioteca de pré-processamento C

`libgcc` Contém suporte de tempo de execução para **gcc**

`libgcov` Essa biblioteca é vinculada a um aplicativo quando GCC for instruído a habilitar criação de perfil

`libgomp` Implementação GNU da API OpenMP para programação paralela de memória compartilhada multiplataforma em C/C++ e Fortran

`libitm` A biblioteca de memória transacional GNU

`liblsan` A biblioteca de tempo de execução do Leak Sanitizer

`liblto_plugin`	Plugin LTO do GCC permite ao binutils processar arquivos objeto produzidos por GCC com LTO habilitado
`libquadmath`	API da Biblioteca Matemática de Precisão Quádrupla GCC
`libssp`	Contém rotinas que suportam a funcionalidade de proteção contra esmagamento de pilha do GCC
`libstdc++`	A biblioteca C++ padrão
`libstdc++fs`	Biblioteca de Sistema de Arquivos ISO/IEC TS 18822:2015
`libsupc++`	Fornece rotinas de suporte para a linguagem de programação C++
`libtsan`	A biblioteca de tempo de execução do Thread Sanitizer
`libubsan`	A biblioteca de tempo de execução do Undefined Behavior Sanitizer

8.27. Pkg-config-0.29.2

O pacote pkg-config contém uma ferramenta para passar o caminho include e (ou) caminhos de biblioteca para ferramentas de construção durante as fases configure e make de instalações de pacote.

Tempo aproximado de construção: 0,3 UPC
Espaço em disco exigido: 29 MB

8.27.1. Instalação de Pkg-config

Prepare Pkg-config para compilação:

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

O significado das novas opções de configure:

`--with-internal-glib`
Isso permitirá que pkg-config use a versão interna dele de Glib, pois uma versão externa não está disponível em LFS.

`--disable-host-tool`
Essa opção desabilita a criação de um indesejado hard link para o aplicativo pkg-config.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.27.2. Conteúdo de Pkg-config

Aplicativo instalado: pkg-config
Diretório instalado: /usr/share/doc/pkg-config-0.29.2

Descrições Curtas

pkg-config Retorna meta informação para a biblioteca ou pacote especificada

8.28. Ncurses-6.3

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

Tempo aproximado de construção: 0,4 UPC
Espaço em disco exigido: 45 MB

8.28.1. Instalação de Ncurses

Prepare Ncurses para compilação:

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

O significado das novas opções de configure:

`--enable-widec`
Essa chave faz com que bibliotecas de caracteres largos (por exemplo, `libncursesw.so.6.3`) sejam construídas em vez de bibliotecas normais (por exemplo, `libncurses.so.6.3`). Essas bibliotecas de caracteres largos são utilizáveis tanto em locales de múltiplos bytes quanto em tradicionais de oito (08) bits, enquanto bibliotecas normais funcionam adequadamente só em locales de oito (08) bits. Bibliotecas de caracteres largos e bibliotecas normais são compatíveis em fonte, mas não são compatíveis em binário.

`--enable-pc-files`
Essa chave gera e instala arquivos .pc para pkg-config.

`--without-normal`
Essa chave desabilita a construção e instalação da maioria das bibliotecas estáticas.

Compile o pacote:

```
make
```

Esse pacote tem uma suíte de teste, entretanto ela só pode ser executada após o pacote ter 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.6.3` no local. Isso talvez quebre o processo de shell que está usando código e dados a partir do arquivo de biblioteca. Instale o pacote com `DESTDIR`, e substitua o arquivo de biblioteca corretamente usando comando **install**. Um arquivamento estático inútil que não é manejado por **configure** também é removido:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.3 /usr/lib
rm -v  dest/usr/lib/{libncursesw.so.6.3,libncurses++w.a}
cp -av dest/* /
```

Muitos aplicativos ainda esperam que o vinculador seja capaz de encontrar bibliotecas Ncurses de caracteres não largos. Ajuste tais aplicativos para vincularem com bibliotecas de caracteres largos por meio de links simbólicos e scripts de vinculador:

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
done
```

Finalmente, certifique-se de que aplicativos antigos que procuram por `-lcurses` em tempo de construção ainda sejam construíveis:

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

Se desejado, então instale a documentação do Ncurses:

```
mkdir -pv      /usr/share/doc/ncurses-6.3
cp -v -R doc/* /usr/share/doc/ncurses-6.3
```

Nota

As instruções acima não criam bibliotecas Ncurses de caracteres não largos, uma vez que nenhum pacote instalado por compilação a partir de 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 versão 5. Se você precisa ter tais bibliotecas, por causa de algum aplicativo somente binário ou para estar conforme com 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
```

8.28.2. Conteúdo de Ncurses

Aplicativos instalados: captoinfo (link para tic), clear, infocmp, infotocap (link para tic), ncursesw6-config, reset (link para tset), tabs, tic, toe, tput, e tset

Bibliotecas instaladas: libcursesw.so (link simbólico e script de vinculador para libncursesw.so), libformw.so, libmenuw.so, libncursesw.so, libpanelw.so, e homônimos delas de caractere não largo sem "w" nos nomes de biblioteca.

Diretórios instalados: /usr/share/tabset, /usr/share/terminfo, e /usr/share/doc/ncurses-6.3

Descrições Curtas

captoinfo	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	Fornece informação de configuração para 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 terminfo que traduz um arquivo terminfo do formato fonte para o formato binário necessário para as rotinas de biblioteca ncurses [Um arquivo terminfo contém informação sobre as capacidades 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 capacidades dependentes de terminal disponíveis para o shell; também pode ser usado para reconfigurar ou inicializar um terminal ou reportar o nome longo dele
tset	Pode ser usado para inicializar terminais
`libcursesw`	Um link para `libncursesw`
`libncursesw`	Contém funções para exibir texto em muitas formas complexas em uma tela de terminal; um bom exemplo do uso dessas funções é o menu exibido durante o **make menuconfig** do kernel
`libformw`	Contém funções para implementar formulários
`libmenuw`	Contém funções para implementar menus
`libpanelw`	Contém funções para implementar painéis

8.29. Sed-4.8

O pacote Sed contém um editor de fluxo.

Tempo aproximado de construção: 0,4 UPC
Espaço em disco exigido: 31 MB

8.29.1. Instalação de Sed

Prepare Sed para compilação:

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

Compile o pacote e gere a documentação HTML:

```
make
make html
```

Para testar os resultados, execute:

```
chown -Rv tester .
su tester -c "PATH=$PATH make check"
```

Instale o pacote e documentação dele:

```
make install
install -d -m755           /usr/share/doc/sed-4.8
install -m644 doc/sed.html /usr/share/doc/sed-4.8
```

8.29.2. Conteúdo do Sed

Aplicativo instalado: sed
Diretório instalado: /usr/share/doc/sed-4.8

Descrições Curtas

sed Filtra e transforma arquivos de texto em uma passagem única

8.30. Psmisc-23.4

O pacote Psmisc contém aplicativos para mostrar informação sobre processos em execução.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 5,6 MB

8.30.1. Instalação do Psmisc

Prepare Psmisc para compilação:

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

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

8.30.2. Conteúdo do Psmisc

Aplicativos instalados: fuser, killall, peekfd, prtstat, pslog, pstree, e pstree.x11 (link para pstree)

Descrições Curtas

fuser Reporta 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 sobre um processo

pslog Reporta 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

8.31. Gettext-0.21

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 da(o) usuária(o).

Tempo aproximado de construção: 2,7 UPC
Espaço em disco exigido: 233 MB

8.31.1. Instalação do Gettext

Prepare Gettext para compilação:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.21
```

Compile o pacote:

```
make
```

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

```
make check
```

Instale o pacote:

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

8.31.2. 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-0.21, /usr/share/gettext, e /usr/share/gettext-0.19.8

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 de idioma natural para o idioma da(o) usuária(o) procurando a tradução em um catálogo de mensagens
gettext.sh Primariamente serve como uma biblioteca de função de shell para gettext
gettextize Copia todos os arquivos Gettext padrão 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 funde os arquivos `.po` fornecidos
msgcmp	Compara dois arquivos `.po` para verificar se ambos contém o mesmo conjunto de sequências de caracteres de msgid
msgcomm	Encontra as mensagens que são comuns aos arquivos `.po` fornecidos
msgconv	Converte um catálogo de tradução para uma codificação de caracteres diferente
msgen	Cria um catálogo de tradução em inglês
msgexec	Aplica um comando a todas as traduções de um catálogo de tradução
msgfilter	Aplica um filtro a todas as traduções de um catálogo de tradução
msgfmt	Gera um catálogo de mensagem binária a partir de um catálogo de tradução
msggrep	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 `.po`, inicializando a meta informação com valores oriundos do ambiente da(o) usuária(o)
msgmerge	Combina duas traduções cruas em um arquivo único
msgunfmt	Descompila um catálogo de mensagem binário em um 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	Recodifica 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
`libasprintf`	define a classe *autosprintf*, que torna as rotinas de saída formatada em C utilizáveis em aplicativos C++, para uso com as sequências de caracteres *<string>* e os fluxos *<iostream>*
`libgettextlib`	uma biblioteca privada contendo rotinas comuns usadas pelos vários aplicativos Gettext; elas não são destinadas para uso geral
`libgettextpo`	Usado para escrever aplicativos especializados que processam arquivos `.po`; essa biblioteca é usada quando os aplicativos padrão fornecidos com Gettext (tais como **msgcomm**, **msgcmp**, **msgattrib**, e **msgen**) não são suficientes
`libgettextsrc`	Uma biblioteca privada contendo rotinas comuns usadas pelos vários aplicativos Gettext; elas não são destinadas para uso geral
`libtextstyle`	Biblioteca de estilo de texto
`preloadable_libintl`	Uma biblioteca, destinada a ser usada por LD_PRELOAD que auxilia `libintl` no registro de mensagens não traduzidas

8.32. Bison-3.8.2

O pacote Bison contém um gerador de analisador.

Tempo aproximado de construção: 6,3 UPC
Espaço em disco exigido: 53 MB

8.32.1. Instalação do Bison

Prepare 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), execute:

```
make check
```

Instale o pacote:

```
make install
```

8.32.2. 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

8.33. Grep-3.7

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

Tempo aproximado de construção: 0,9 UPC
Espaço em disco exigido: 36 MB

8.33.1. Instalação do Grep

Prepare Grep para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.33.2. Conteúdo do Grep

Aplicativos instalados: egrep, fgrep, e grep

Descrições Curtas

egrep Imprime linhas que correspondem a uma expressão regular estendida

fgrep Imprime linhas que correspondem a uma lista de sequências de caracteres fixas

grep Imprime linhas que correspondem a expressão regular básica

8.34. Bash-5.1.16

O pacote Bash contém o Bourne-Again SHell.

Tempo aproximado de construção: 1,5 UPC
Espaço em disco exigido: 50 MB

8.34.1. Instalação do Bash

Prepare Bash para compilação:

```
./configure --prefix=/usr                      \
            --docdir=/usr/share/doc/bash-5.1.16 \
            --without-bash-malloc              \
            --with-installed-readline
```

O significado da nova opção de configure:

`--with-installed-readline`
Essa opção diz a Bash para usar a biblioteca `readline` que já está instalada no sistema em vez de usar sua própria versão de readline.

Compile o pacote:

```
make
```

Pule para “Instale o pacote” se não executar a suíte de teste.

Para preparar os testes, garanta que a(o) usuária(o) `tester` pode escrever na árvore de fontes:

```
chown -Rv tester .
```

A suíte de teste do pacote é desenhada para ser executada como uma(m) usuária(o) não root que é proprietária(o) do terminal conectado à entrada padrão. Para satisfazer a exigência, crie um novo pseudo terminal usando Expect e execute os testes como a(o) usuária (o) `tester`:

```
su -s /usr/bin/expect tester << EOF
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

Instale o pacote:

```
make install
```

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

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

8.34.2. Conteúdo do Bash

Aplicativos instalados: bash, bashbug, e sh (link para bash)
Diretórios instalados: /usr/include/bash, /usr/lib/bash, e /usr/share/doc/bash-5.1.16

Descrições Curtas

bash Um interpretador de comandos vastamente utilizado; 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 a(o) usuária(o) a compor e enviar relatórios de defeitos formatados padrão concernentes a **bash**

sh Um link simbólico para o aplicativo **bash**; quando invocado como **sh**, **bash** tenta imitar o comportamento de inicialização de versões históricas do **sh** o mais próximo possível, enquanto também conformando com o padrão POSIX

8.35. Libtool-2.4.6

O pacote Libtool contém o script de suporte à biblioteca genérica GNU. Ele esconde a complexidade de usar bibliotecas compartilhadas em uma interface consistente e portável.

Tempo aproximado de construção: 1,5 UPC
Espaço em disco exigido: 43 MB

8.35.1. Instalação do Libtool

Prepare Libtool para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Nota

O tempo de teste para libtool pode ser reduzido significativamente em um sistema com múltiplos núcleos. Para fazer isso, acrescente **TESTSUITEFLAGS=-j<N>** ao final da linha acima. Por exemplo, usar -j4 pode reduzir o tempo de teste em mais que 60 por cento.

Cinco testes são conhecidos por falharem dentro do ambiente de construção LFS devido a uma dependência circular, porém todos os testes passam se verificados novamente após automake ser instalado.

Instale o pacote:

```
make install
```

Remova uma biblioteca estática inútil:

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

8.35.2. Conteúdo do Libtool

Aplicativos instalados: libtool e libtoolize
Biblioteca instalada: libltdl.so
Diretórios instalados: /usr/include/libltdl e /usr/share/libtool

Descrições Curtas

libtool Fornece serviços generalizados de suporte à construção de bibliotecas

libtoolize Fornece uma maneira padrão de adicionar suporte **libtool** a um pacote

`libltdl` Esconde as várias dificuldades do dlopening de bibliotecas

8.36. GDBM-1.23

O pacote GDBM contém o GNU Database Manager. Ele é uma biblioteca de funções de banco 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 por sua chave e deletar uma chave junto com seus dados.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 13 MB

8.36.1. 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 a construção da biblioteca de compatibilidade libgdbm. Alguns pacotes fora do LFS talvez exijam as rotinas DBM mais antigas que ela fornece.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.36.2. Conteúdo do GDBM

Aplicativos instalados: gdbm_dump, gdbm_load, e gdbmtool
Bibliotecas instaladas: libgdbm.so e libgdbm_compat.so

Descrições Curtas

gdbm_dump Despeja um banco de dados GDBM para um arquivo
gdbm_load Recria um banco de dados GDBM a partir de um arquivo de despejo
gdbmtool Testa e modifica um banco de dados GDBM
`libgdbm` Contém funções para manipular um banco de dados com hash
`libgdbm_compat` Biblioteca de compatibilidade contendo funções DBM mais antigas

8.37. 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: 6,0 MB

8.37.1. 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, execute:

```
make -j1 check
```

Instale o pacote:

```
make install
```

8.37.2. 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

8.38. Expat-2.4.6

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

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 12 MB

8.38.1. Instalação do Expat

Prepare Expat para compilação:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.4.6
```

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

Se desejado, instale a documentação:

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.4.6
```

8.38.2. Conteúdo do Expat

Aplicativo instalado: xmlwf
Biblioteca instalada: libexpat.so
Diretório instalado: /usr/share/doc/expat-2.4.6

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

8.39. Inetutils-2.2

O pacote Inetutils contém aplicativos para redes básicas.

Tempo aproximado de construção: 0,3 UPC
Espaço em disco exigido: 30 MB

8.39.1. Instalação do Inetutils

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 de 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 instale isso, 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 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 confiáveis. Observe que substituições melhores estão disponíveis para muitos desses servidores.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

Mova um aplicativo para o local adequado:

```
mv -v /usr/{,s}bin/ifconfig
```

8.39.2. 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 de protocolo de transferência de arquivos

hostname Relata ou configura o nome do dispositivo

ifconfig Gerencia interfaces de rede

ping Envia pacotes de solicitação de echo e informa quanto tempo as respostas demoram

ping6 Uma versão do **ping** para redes IPv6

talk É usado para conversar com outra(o) usuária(o)

telnet Uma interface para o protocolo TELNET

tftp Um aplicativo de transferência de arquivos trivial

traceroute Rastreia a rota que seus pacotes fazem a partir do dispositivo em que você está trabalhando para outro dispositivo em uma rede, mostrando todos os saltos intermediários (gateways) ao longo do caminho

8.40. Less-590

O pacote Less contém um visualizador de arquivos de texto.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 4,2 MB

8.40.1. Instalação do Less

Prepare Less para compilação:

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

O significado das opções de configure:

`--sysconfdir=/etc`
Essa opção diz aos aplicativos criados pelo pacote para procurarem em `/etc` pelos arquivos de configuração.

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

8.40.2. 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 a(o) usuária(o) role, encontre sequências de caracteres e pule para marcas

lessecho Necessário para expandir metacaracteres, tais como * e *?*, em nomes de arquivos em sistemas Unix

lesskey Usado para especificar os atalhos de tecla para **less**

8.41. Perl-5.34.0

O pacote Perl contém o Practical Extraction and Report Language.

Tempo aproximado de construção: 9,3 UPC
Espaço em disco exigido: 226 MB

8.41.1. Instalação do Perl

Primeiro, aplique uma correção que conserta um problema destacado por versões recentes do gdbm:

```
patch -Np1 -i ../perl-5.34.0-upstream_fixes-1.patch
```

Essa versão do Perl agora constrói os módulos Compress::Raw::Zlib e Compress::Raw::BZip2. Por padrão, Perl usará uma cópia interna dos fontes para a construção. Execute o seguinte comando de modo que Perl usará as bibliotecas instaladas no sistema:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

Para ter controle completo sobre a maneira como 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 está abaixo para usar os padrões que o Perl detecta automaticamente:

```
sh Configure -des                                         \
             -Dprefix=/usr                                \
             -Dvendorprefix=/usr                          \
             -Dprivlib=/usr/lib/perl5/5.34/core_perl      \
             -Darchlib=/usr/lib/perl5/5.34/core_perl      \
             -Dsitelib=/usr/lib/perl5/5.34/site_perl      \
             -Dsitearch=/usr/lib/perl5/5.34/site_perl     \
             -Dvendorlib=/usr/lib/perl5/5.34/vendor_perl  \
             -Dvendorarch=/usr/lib/perl5/5.34/vendor_perl \
             -Dman1dir=/usr/share/man/man1                \
             -Dman3dir=/usr/share/man/man3                \
             -Dpager="/usr/bin/less -isR"                 \
             -Duseshrplib                                 \
             -Dusethreads
```

O significado das opções de configure:

`-Dvendorprefix=/usr`
Isso garante que **perl** saiba como dizer aos pacotes onde eles deveriam instalar módulos perl deles.

`-Dpager="/usr/bin/less -isR"`
Isso garante que **`less`** seja usado em vez de **`more`**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Uma vez que o Groff ainda não está instalado, **Configure** pensa que nós não queremos páginas de manual para o Perl. Emitir esses parâmetros substitui essa decisão.

`-Duseshrplib`
Constrói uma libperl compartilhada necessária para alguns módulos perl.

`-Dusethreads`
Constrói perl com suporte para camadas.

`-Dprivlib,-Darchlib,-Dsitelib,...`
Essas configurações definem onde o Perl está procurando módulos instalados. As(Os) editoras(es) de LFS optaram por colocá-los em uma estrutura de diretórios baseada na versão Major.Minor do Perl (5.34), a qual permite atualizar o Perl para níveis de Patch mais recentes (5.34.0) sem a necessidade de reinstalar todos os módulos novamente.

Compile o pacote:

```
make
```

Para testar os resultados (aproximadamente 11 UPCs), execute:

```
make test
```

Instale o pacote e limpe:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.41.2. Conteúdo do Perl

Aplicativos instalados:	corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.34.0 (hard link para perl), perlbug, perldoc, perlivp, perlthanks (hard link para perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp e zipdetails
Bibliotecas instaladas:	Muitas, as quais não podem ser todas listadas aqui
Diretório instalado:	/usr/lib/perl5

Descrições Curtas

corelist	Um frontend de linha de comando para Module::CoreList
cpan	Interage com o Comprehensive Perl Archive Network (CPAN) a partir da linha de comando
enc2xs	Constrói uma extensão Perl para o módulo Encode a partir tanto de Mapeamentos de Caracteres Unicode quanto de Arquivos de Codificação Tcl
encguess	Advinha o tipo de codificação de um ou vários arquivos
h2ph	Converte arquivos de cabeçalho C `.h` para arquivos de cabeçalho Perl `.ph`
h2xs	Converte arquivos de cabeçalho C `.h` para extensões Perl
instmodsh	Script de shell para examinar módulos Perl instalados, e pode criar um tarball a partir de um módulo instalado
json_pp	Converte dados entre certos formatos de entrada e saída
libnetcfg	Pode ser usado para configurar o módulo Perl `libnet`
perl	Combina algumas das melhores características do C, **sed**, **awk** e **sh** em uma linguagem canivete suíço única

perl5.34.0	Um hard link para **perl**
perlbug	Usado para gerar relatórios de defeitos sobre o Perl, ou módulos que vem como ele, e enviá-los por correio
perldoc	Exibe uma parte da documentação em formato de pod que está incorporada na árvore de instalação do Perl ou em um script Perl
perlivp	O Procedimento de Verificação de Instalação do Perl; pode ser usado para verificar se o Perl e suas bibliotecas foram instalados corretamente
perlthanks	Usado para gerar mensagens de agradecimento para enviar para as(os) desenvolvedoras(es) 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 formatada *roff
pod2text	Converte dados pod para texto ASCII formatado
pod2usage	Imprime mensagens de uso a partir de documentos pod incorporados em arquivos
podchecker	Verifica a sintaxe de arquivos de documentação no formato pod
podselect	Exibe seções selecionadas de 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 de aviso detalhados em Perl
xsubpp	Converte código Perl XS em código C
zipdetails	Exibe detalhes sobre a estrutura interna de um arquivo Zip

8.42. XML::Parser-2.46

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

8.42.1. Instalação do XML::Parser

Prepare XML::Parser para compilação:

```
perl Makefile.PL
```

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make test
```

Instale o pacote:

```
make install
```

8.42.2. Conteúdo do XML::Parser

Módulo instalado: Expat.so

Descrições Curtas

`Expat` Fornece a interface Perl Expat

8.43. 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

8.43.1. Instalação do Intltool

Primeiro, conserte 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, execute:

```
make check
```

Instale o pacote:

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

8.43.2. 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 mescla eles com arquivos de tradução
intltool-update	Atualiza os arquivos de modelo po e mescla eles com as traduções

8.44. Autoconf-2.71

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

Tempo aproximado de construção: menos que 0,1 UPC (cerca de 6,8 UPC com os testes)
Espaço em disco exigido: 24 MB

8.44.1. Instalação do Autoconf

Prepare Autoconf para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Nota

O tempo de teste para autoconf pode ser reduzido significativamente em um sistema com múltiplos núcleos. Para fazer isso, acrescente **TESTSUITEFLAGS=-j<N>** ao final da linha acima. Por exemplo, usar -j4 pode reduzir o tempo de teste em mais que 60 por cento.

Instale o pacote:

```
make install
```

8.44.2. 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 vários 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 *#define* de C para 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 `configure.in` para um pacote de aplicativos; ele examina os arquivos fonte em uma árvore de diretórios, procurando neles por problemas de portabilidade comuns, e cria

um arquivo `configure.scan` que serve como um arquivo `configure.in` preliminar para o pacote

autoupdate Modifica um arquivo `configure.in` que ainda chama macros **autoconf** por seus nomes antigos para usar os nomes de macro atuais

ifnames Ajuda ao escrever arquivos `configure.in` para um pacote de aplicativos; ele imprime os identificadores que o pacote usa em condicionais de preprocessador C [Se um pacote já foi configurado para ter alguma portabilidade, então esse aplicativo pode ajudar a determinar o que **configure** precisa checar. Ele também pode preencher lacunas em um arquivo `configure.in` gerado por **autoscan**].

8.45. Automake-1.16.5

O pacote Automake contém aplicativos para gerar Makefiles para uso com Autoconf.

Tempo aproximado de construção: menos que 0,1 UPC (cerca de 8,3 UPC com os testes)
Espaço em disco exigido: 115 MB

8.45.1. Instalação do Automake

Prepare Automake para compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.16.5
```

Compile o pacote:

```
make
```

Usar a opção de make -j4 acelera os testes, mesmo em sistemas com apenas um processador, devido a atrasos internos em testes individuais. Para testar os resultados, execute:

```
make -j4 check
```

O teste t/subobj.sh é conhecido por falhar.

Instale o pacote:

```
make install
```

8.45.2. Conteúdo do Automake

Aplicativos instalados: aclocal, aclocal-1.16 (hard link com aclocal), automake e automake-1.16 (hard link com automake)
Diretórios instalados: /usr/share/aclocal-1.16, /usr/share/automake-1.16 e /usr/share/doc/automake-1.16.5

Descrições Curtas

aclocal Gera arquivos `aclocal.m4` baseados no conteúdo dos arquivos `configure.in`

aclocal-1.16 Um hard link para **aclocal**

automake Uma ferramenta para gerar automaticamente arquivos `Makefile.in` a partir de arquivos `Makefile.am` [Para criar todos os arquivos `Makefile.in` para um pacote, execute esse aplicativo no diretório superior. Escaneando o arquivo `configure.in`, ele automaticamente encontra cada arquivo `Makefile.am` apropriado e gera o arquivo `Makefile.in` correspondente].

automake-1.16 Um hard link para **automake**

8.46. OpenSSL-3.0.1

O pacote OpenSSL contém ferramentas de gerenciamento e bibliotecas relacionadas à criptografia. Essas são úteis para fornecer funções criptográficas para outros pacotes, tais como OpenSSH, aplicativos de correio eletrônico e navegadores de rede (para acessar sítios HTTPS).

Tempo aproximado de construção: 5,4 UPC
Espaço em disco exigido: 474 MB

8.46.1. 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, execute:

```
make test
```

Um teste, 30-test_afalg.t, é conhecido por falhar em algumas configurações de kernel (dependendo de valores inconsistentes de configurações CONFIG_CRYPTO_USER_API*). Se ele falhar, então ele pode seguramente ser 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-3.0.1
```

Se desejado, instale alguma documentação adicional:

```
cp -vfr doc/* /usr/share/doc/openssl-3.0.1
```

Nota

Você deveria atualizar OpenSSL quando uma versão nova que conserta vulnerabilidades for anunciada. Os lançamentos ocorrem em séries, com uma letra para cada lançamento após o lançamento inicial (por exemplo, 1.1.1, 1.1.1a, 1.1.1b, etc). Por causa de que LFS instala somente as bibliotecas compartilhadas, não existe necessidade de recompilar pacotes que se vinculem a `libcrypto.so` ou `libssl.so` *quando atualizar na mesma série.*

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.

8.46.2. 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-3.0.1

Descrições Curtas

c_rehash é um script Perl que escaneia todos os arquivos em um diretório e adiciona links simbólicos para os valores de hash deles

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 **man 1 openssl**

`libcrypto.so` implementa um intervalo amplo de algoritmos criptográficos usados em vários padrões da Internet. Os serviços fornecidos por essa biblioteca são usados pelas implementações OpenSSL do SSL, TLS e S/MIME e eles também tem sido usados para implementar OpenSSH, OpenPGP e outros padrões criptográficos

`libssl.so` implementa o protocolo Transport Layer Security (TLS v1). Ela fornece uma API rica, documentação sobre a qual pode ser encontrada executando **man 3 ssl**

8.47. Kmod-29

O pacote Kmod contém bibliotecas e utilitários para carregar módulos de kernel

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 12 MB

8.47.1. Instalação do Kmod

Prepare Kmod para compilação:

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

O significado das opções de configure:

`--with-openssl`
Essa opção habilita Kmod a lidar com assinaturas PKCS7 para módulos de kernel.

`--with-xz`, *`--with-zlib`*, e *`--with-zstd`*
Essas opções habilitam Kmod a lidar com módulos de kernel comprimidos.

Compile o pacote:

```
make
```

A suíte de teste desse pacote exige cabeçalhos de kernel crus (não os cabeçalhos de kernel “sanitizados” instalados anteriormente), os quais estão além do escopo do LFS.

Instale o pacote e crie links simbólicos para compatibilidade com Module-Init-Tools (o pacote que anteriormente lidava com módulos de kernel do Linux):

```
make install

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

ln -sfv kmod /usr/bin/lsmod
```

8.47.2. 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 encontra no conjunto existente de módulos; esse arquivo de dependência é usado por **modprobe** para carregar automaticamente os módulos exigidos

insmod Instala um módulo carregável no kernel em execução

kmod Carrega e descarrega módulos de kernel

lsmod Lista módulos atualmente carregados

modinfo Examina um arquivo objeto associado com um módulo de kernel e exibe qualquer informação que ele possa coletar

modprobe Usa um arquivo de dependência, criado por **depmod**, para carregar automaticamente módulos relevantes

rmmod Descarrega módulos a partir do kernel em execução

`libkmod` Essa biblioteca é usada por outros aplicativos para carregar e descarregar módulos de kernel

8.48. Libelf oriundo de Elfutils-0.186

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

Tempo aproximado de construção: 0,9 UPC
Espaço em disco exigido: 116 MB

8.48.1. Instalação do Libelf

Libelf é parte do pacote elfutils-0.186. Use o elfutils-0.186.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, execute:

```
make check
```

Instale apenas Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.48.2. Conteúdo do Libelf

Biblioteca instalada: libelf.so (link simbólico) e libelf-0.186.so
Diretório instalado: /usr/include/elfutils

Descrições Curtas

`libelf` Contém funções de API para lidar com arquivos objeto ELF

8.49. Libffi-3.4.2

A biblioteca Libffi fornece uma interface de programação portável e de alto nível para várias convenções de chamada. Isso permite a uma(m) programadora(r) chamar qualquer função especificada por uma descrição de interface de chamada em tempo de execução.

Tempo aproximado de construção: 1,9 UPC
Espaço em disco exigido: 10 MB

8.49.1. Instalação do Libffi

Nota

Semelhante a GMP, libffi constrói com otimizações específicas para o processador em uso. Se construir para outro sistema, então exporte CFLAGS e CXXFLAGS para especificar uma construção genérica para sua arquitetura. Se isso não for feito, então todos os aplicativos que se vincularem a libffi deflagrarão Illegal Operation Errors.

Prepare libffi para compilação:

```
./configure --prefix=/usr          \
            --disable-static       \
            --with-gcc-arch=native \
            --disable-exec-static-tramp
```

O significado da opção de configure:

`--with-gcc-arch=native`
Garante que GCC otimiza para o sistema atual. Se isso não for especificado, então o sistema é presumido e o código gerado talvez não esteja correto para alguns sistemas. 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 de sistema alternativos, veja-se *as opções de x86 no manual do GCC*.

`--disable-exec-static-tramp`
Desabilita o suporte de trampolim estático. Ele é uma nova característica de segurança em libffi, porém alguns pacotes BLFS (notadamente GJS e gobject-introspection) não foram adaptados para ele.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.49.2. Conteúdo do Libffi

Biblioteca instalada: libffi.so

Descrições Curtas

`libffi` contém as funções da API da interface de função estrangeira

8.50. Python-3.10.2

O pacote Python 3 contém o ambiente Python de desenvolvimento. Ele é útil para programação orientada a objetos, escrita de scripts, prototipagem de aplicativos grandes, ou desenvolvimento de aplicações inteiras.

Tempo aproximado de construção: 4,3 UPC
Espaço em disco exigido: 275 MB

8.50.1. Instalação do Python 3

Prepare Python para compilação:

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

O significado das opções de configure:

`--with-system-expat`
Essa chave habilita vinculação contra a versão de sistema do Expat.

`--with-system-ffi`
Essa chave habilita vinculação contra a versão de sistema de libffi.

`--with-ensurepip=yes`
Essa chave habilita construir os aplicativos de empacotamento **pip** e **setuptools**.

`--enable-optimizations`
Essa chave habilita otimizações estáveis, porém onerosas.

Compile o pacote:

```
make
```

Executar os testes neste ponto não é recomendado. Os testes são conhecidos por travar indefinidamente dentro de um ambiente LFS parcial. Se desejado, então os testes podem ser reexecutados ao final deste capítulo ou quando Python 3 for reinstalado em BLFS. Para executar os testes de qualquer maneira, emita **make test**.

Instale o pacote:

```
make install
```

Se desejado, então instale a documentação pré-formatada:

```
install -v -dm755 /usr/share/doc/python-3.10.2/html

tar --strip-components=1  \
    --no-same-owner       \
    --no-same-permissions \
    -C /usr/share/doc/python-3.10.2/html \
    -xvf ../python-3.10.2-docs-html.tar.bz2
```

O significado dos comandos de instalação de documentação:

`--no-same-owner` e `--no-same-permissions`
Garanta que os arquivos instalados tenham a propriedade e as permissões corretas. Sem essas opções, usar tar instalará os arquivos de pacote com os valores da(o) criadora(r) upstream.

8.50.2. Conteúdo do Python 3

Aplicativos instalados: 2to3, idle3, pip3, pydoc3, python3 e python3-config
Bibliotecas instaladas: libpython3.10.so e libpython3.so
Diretórios instalados: /usr/include/python3.10, /usr/lib/python3 e /usr/share/doc/python-3.10.2

Descrições Curtas

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

idle3 é um script encapsulador que abre um editor GUI ciente de Python. Para esse script executar, você precisa ter instalado Tk antes do Python, de forma que o módulo Tkinter 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 Python

python3 é uma linguagem de programação orientada a objeto, interativa e interpretada

8.51. Ninja-1.10.2

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

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 64 MB

Dica

Esta seção não é estritamente exigida para LFS se não usar systemd. Por outro lado, ninja associado a meson fazem uma combinação de sistema de construção poderosa, o qual é esperado que seja usado mais e mais frequentemente. Ele é exigido por muitos pacotes no *livro BLFS*.

8.51.1. Instalação do Ninja

Quando executado, ninja normalmente executa um número máximo de processos em paralelo. Por padrão, esse é o número de núcleos no sistema mais dois. Em alguns casos, isso pode superaquecer uma CPU ou deixar o sistema sem memória. Se executar a partir da linha de comando, então passar um parâmetro -jN limitará o número de processos paralelos, porém alguns pacotes embutem a execução de ninja e não passam um parâmetro -j.

Usar o procedimento *opcional* abaixo permite que uma(m) usuária(o) 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 adicione a capacidade de usar a variável de ambiente NINJAJOBS executando:

```
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.

Para testar os resultados, execute:

```
./ninja ninja_test
./ninja_test --gtest_filter=-SubprocessTest.SetWithLots
```

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
```

8.51.2. Conteúdo do Ninja

Aplicativo instalado: ninja

Descrições Curtas

ninja é o sistema de construção Ninja

8.52. Meson-0.61.1

Meson é um sistema de construção de código fonte aberto destinado para ser ambos extremamente rápido e tão amigável à(ao) usuária(o) quanto possível.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 41 MB

Dica

Esta seção não é estritamente exigida para LFS se não usar systemd. Por outro lado, meson/ninja é um sistema de construção poderoso, o qual é esperado que seja usado mais e mais frequentemente. Ele é exigido por muitos pacotes no *livro BLFS*.

8.52.1. Instalação do Meson

Compile Meson com o seguinte comando:

```
python3 setup.py build
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
python3 setup.py install --root=dest
cp -rv dest/* /
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/compl
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_m
```

O significado dos parâmetros de install:

`--root=dest`
Por padrão, **python3 setup.py install** instala vários arquivos (tais como páginas de manual) em Python Eggs. Com um local raiz especificado, **setup.py** instala esses arquivos na hierarquia padrão. Então a hierarquia pode apenas ser copiada para o local padrão.

8.52.2. Conteúdo do Meson

Aplicativo instalado: meson
Diretórios instalados: /usr/lib/python3.10/site-packages/meson-0.61.1-py3.10.egg-info e /usr/lib/python3.10/site-packages/mesonbuild

Descrições Curtas

meson Um sistema de construção de alta produtividade

8.53. Coreutils-9.0

O pacote Coreutils contém utilitários para mostrar e configurar as características básicas de sistema.

Tempo aproximado de construção: 2,6 UPC
Espaço em disco exigido: 153 MB

8.53.1. Instalação do Coreutils

POSIX exige que aplicativos originários do Coreutils reconheçam limites de carácter corretamente mesmo em locales multibyte. A seguinte correção conserta essa não-conformidade e outros defeitos relacionados à internacionalização.

```
patch -Np1 -i ../coreutils-9.0-i18n-1.patch
```

Nota

No passado, muitos defeitos foram encontrados nessa correção. Quando reportar novos defeitos para as(os) mantenedoras(es) do Coreutils, por favor verifique primeiro se eles são reproduzíveis sem essa correção.

Agora, conserte um problema com valores de retorno de chmod:

```
patch -Np1 -i ../coreutils-9.0-chmod_fix-1.patch
```

Agora prepare Coreutils para compilação:

```
autoreconf -fiv
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

O significado das opções de configure:

autoreconf
A correção para internacionalização modificou o sistema de construção do pacote, então os arquivos de configuração tem de ser regenerados.

`FORCE_UNSAFE_CONFIGURE=1`
Essa variável de ambiente permite que o pacote seja construído como a(o) usuária(o) `root`.

`--enable-no-install-program=kill,uptime`
O propósito dessa chave é o de impedir que o Coreutils instale binários que serão instalados por outros pacotes posteriormente.

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 usuária(o) `root`:

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

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

```
echo "dummy:x:102:tester" >> /etc/group
```

Conserte algumas das permissões de modo que a(o) usuária(o) não-root possa compilar e executar os testes:

```
chown -Rv tester .
```

Agora execute os testes:

```
su tester -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

O teste test-getlogin é conhecido por falhar dentro do ambiente chroot do LFS.

Remova o grupo temporário:

```
sed -i '/dummy/d' /etc/group
```

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
```

8.53.2. 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/[, que 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 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 fazer ou um número octal representando as novas permissões

chown Muda a propriedade da(o) usuária(o) e (ou) 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, exibindo 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 exibindo 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 hora atual no formato dado, ou configura a data do sistema

dd Copia um arquivo usando o tamanho de bloco e contagem dados, enquanto opcionalmente realiza conversões sobre ele

df Reporta a quantidade de espaço de disco disponível (e usada) em todos os sistemas de arquivos montados, ou apenas nos sistemas de arquivos contendo os arquivos selecionados

dir Lista o conteúdo de cada diretório dado (o mesmo que o comando **ls**)

dircolors Gera comandos para configurar a variável de ambiente `LS_COLOR` para mudar o esquema de cores usado por **ls**

dirname Remove o sufixo que não é diretório de um nome de arquivo

du Relata a quantidade de espaço de disco usado pelo diretório atual, por cada diretório dado (incluindo todos subdiretórios) ou por cada um 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 de todos os números 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 Relata relacionamentos de membro de grupo de uma(m) usuária(o)

head Imprime as primeiras dez linhas (ou o número de linhas dado) de cada arquivo dado

hostid Relata o número identificador (em hexadecimal) do dispositivo

id Relata o efetivo ID de usuária(o), ID de grupo, e os relacionamentos de membro de grupo da(o) usuária(o) atual ou usuária(o) especificada(o)

install	Copia arquivos enquanto configura seus modos de permissão e, se possível, seus proprietário e grupo
join	Junta as linhas que tem idênticos campos de junção a partir de dois arquivos separados
link	Cria um hard link com o nome dado para um arquivo
ln	Faz hard links ou soft (simbólico) links entre arquivos
logname	Relata o nome de login da(o) usuária(o) atual
ls	Lista o conteúdo de cada diretório dado
md5sum	Relata ou verifica somas de verificação Message Digest 5 (MD5)
mkdir	Cria diretórios com os nomes dados
mkfifo	Cria First-In, First-Outs (FIFOs), um "pipe 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 scripts
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 sua saída redirecionada para um arquivo de registro
nproc	Imprime o número de unidades de processamento disponíveis para um processo
numfmt	Converte números para ou 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 relata algumas informações sobre as(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	Relata o nome do diretório de trabalho atual
readlink	Relata 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 relata 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 padrão enquanto escreve tanto para saída 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 marcas temporais 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 padrão
true	Não faz nada, com sucesso; sempre sai com um código de status indicando sucesso
truncate	Comprime 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	Relata o nome de arquivo do terminal conectado à entrada padrão
uname	Relata informação de sistema
unexpand	Converte espaços para tabulação
uniq	Descarta todas, exceto uma das sucessivas linhas idênticas
unlink	Remove o arquivo dado
users	Relata os nomes das(os) usuárias(os) atualmente logados
vdir	É o mesmo que **ls -l**
wc	Relata o número de linhas, palavras e bytes para cada arquivo dado, assim como uma linha de total quando mais que um arquivo for dado
who	Relata quem está logado

whoami	Relata o nome de usuária(o) associado com o ID de usuária(o) efetivo atual
yes	Repetidamente retorna “y” ou uma sequência de caracteres dada até que seja terminado
`libstdbuf`	Biblioteca usada por **stdbuf**

8.54. Check-0.15.2

Check é uma estrutura de teste de unidade para C.

Tempo aproximado de construção: 0,1 UPC (cerca de 3,8 UPC com os testes)
Espaço em disco exigido: 12 MB

8.54.1. Instalação do Check

Prepare Check para compilação:

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

Construa o pacote:

```
make
```

Compilação agora está completa. Para executar a suíte de teste do Check, execute o seguinte comando:

```
make check
```

Instale o pacote:

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

8.54.2. 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.{a,so}` Contém funções que permitem que Check seja chamado a partir de um aplicativo de teste

8.55. Diffutils-3.8

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

Tempo aproximado de construção: 0,6 UPC
Espaço em disco exigido: 34 MB

8.55.1. Instalação do Diffutils

Prepare Diffutils para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.55.2. Conteúdo do Diffutils

Aplicativos instalados: cmp, diff, diff3 e sdiff

Descrições Curtas

cmp Compara dois arquivos e relata se ou em quais bytes eles diferem
diff Compara dois arquivos ou diretórios e relata quais linhas nos arquivos diferem
diff3 Compara três arquivos linha por linha
sdiff Mescla dois arquivos e interativamente exibe os resultados

8.56. Gawk-5.1.1

O pacote Gawk contém aplicativos para manipular arquivos de texto.

Tempo aproximado de construção:	0,4 UPC
Espaço em disco exigido:	43 MB

8.56.1. Instalação do Gawk

Primeiro, garanta que alguns arquivos desnecessários não sejam instalados:

```
sed -i 's/extras//' Makefile.in
```

Prepare Gawk para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

Se desejado, então instale a documentação:

```
mkdir -pv                                   /usr/share/doc/gawk-5.1.1
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-5.1.1
```

8.56.2. Conteúdo do Gawk

Aplicativos instalados:	awk (link para gawk), gawk e awk-5.1.1
Bibliotecas instaladas:	filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.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-5.1.1

Descrições Curtas

awk	Um link para **gawk**
gawk	Um aplicativo para manipular arquivos de texto; é a implementação GNU do **awk**
gawk-5.1.1	Um hard link para **gawk**

8.57. Findutils-4.9.0

O pacote Findutils contém aplicativos para procurar arquivos. Esses aplicativos são fornecidos para procurar recursivamente dentro de uma árvore de diretório e para criar, manter e buscar um banco de dados (geralmente mais rápido que o find recursivo, porém não é confiável se o banco de dados não for atualizado recentemente).

Tempo aproximado de construção: 0,9 UPC
Espaço em disco exigido: 51 MB

8.57.1. Instalação do Findutils

Prepare Findutils para compilação:

```
case $(uname -m) in
    i?86)   TIME_T_32_BIT_OK=yes ./configure --prefix=/usr --localstatedir=/var/li
    x86_64) ./configure --prefix=/usr --localstatedir=/var/lib/locate ;;
esac
```

O significado das opções de configure:

TIME_32_BIT_OK=yes
Essa configuração é necessária para construir em um sistema de 32 bits.

`--localstatedir`
Essa opção muda o local da base de dados **locate** para estar em `/var/lib/locate`, o qual é conforme com FHS.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
chown -Rv tester .
su tester -c "PATH=$PATH make check"
```

Instale o pacote:

```
make install
```

8.57.2. Conteúdo do Findutils

Aplicativos instalados: find, locate, updatedb e xargs
Diretório instalado: /var/lib/locate

Descrições Curtas

find Pesquisa em árvores de diretórios dadas por arquivos correspondendo a critérios especificados

locate Pesquisa em um banco de dados de nomes de arquivo e relata os nomes que contém uma sequência de caracteres dada ou correspondem a um padrão dado

updatedb Atualiza o banco 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 no banco de dados

xargs Pode ser usado para aplicar um comando dado a uma lista de arquivos

8.58. Groff-1.22.4

O pacote Groff contém aplicativos para processar e formatar texto.

Tempo aproximado de construção: 0,5 UPC
Espaço em disco exigido: 88 MB

8.58.1. Instalação do Groff

Groff espera que a variável de ambiente PAGE contenha o tamanho de papel padrão. Para usuárias(os) nos Estados Unidos da América do Norte, *PAGE=letter* é apropriado. Em outros lugares, *PAGE=A4* talvez seja mais adequado. Embora o tamanho do papel padrão 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=<paper_size> ./configure --prefix=/usr
```

Esse pacote não suporta construção paralela. Compile o pacote:

```
make -j1
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

8.58.2. 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-1.22.4 e /usr/share/groff

Descrições Curtas

addftinfo Lê um arquivo de fonte troff e adiciona algumas informações de métrica de fonte adicionais que são usadas 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 troff em comandos que são entendidos por troff

eqn2graph Converte uma EQN (equação) 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 adição do código perl em arquivos groff

gpinyin	Preprocessador para groff, permitindo adição do idioma semelhante a Chinês Europeu Pinyin em arquivos groff
grap2graph	Converte um diagrama grap em uma imagem de bitmap recortada
grn	Um preprocessador **groff** para arquivos gremlin
grodvi	Um controlador para **groff** que produz formato dvi do TeX
groff	Um frontal para o sistema de formatação de documentos groff; normalmente, 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 **groff** `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s` e `-t` são exigidas para imprimir arquivos, e relata o 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 no formato PCL5 adequado para uma impressora HP LaserJet 4
gropdf	Traduz a saída do GNU **troff** para PDF
grops	Traduz a saída do GNU **troff** para PostScript
grotty	Traduz a saída 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 os bancos de dados bibliográficos com um arquivo especificado para uso com **refer**, **lookbib** e **lkbib**
lkbib	Pesquisa em bancos de dados bibliográficos por referências que contenham chaves especificadas e relata quaisquer referências encontradas
lookbib	Imprime um prompt na saída de erro padrão (a não ser que a entrada padrão não seja um terminal); lê uma linha contendo um conjunto de palavras chave a partir da entrada padrão; pesquisa em bancos de dados bibliográficos, em um arquivo especificado, por referências contendo aquelas palavras chave; imprime quaisquer referências encontradas na saída padrão; e repete esse processo até o final da entrada
mmroff	Um preprocessador simples para **groff**
neqn	Formata equações para saída American Standard Code for Information Interchange (ASCII)
nroff	Um script que emula o comando **nroff** usando **groff**
pdfmom	É um encapsulador em torno de groff que facilita a produção de documentos PDF a partir de arquivos formatados com as macros mom
pdfroff	Cria documentos pdf usando groff
pfbtops	Traduz uma fonte PostScript em formato `.pfb` para ASCII
pic	Compila descrições de imagens embutidas em arquivos de entrada troff ou TeX em comandos entendidos por TeX ou **troff**
pic2graph	Converte um diagrama PIC em uma imagem recortada
post-grohtml	Traduz a saída do GNU **troff** para HTML

preconv	Converte codificação de arquivos de entrada em alguma coisa que o GNU **troff** entende
pre-grohtml	Traduz a saída do GNU **troff** para HTML
refer	Copia o conteúdo de um arquivo para a saída padrão, exceto aquelas linhas entre *.[* e *.]* que são interpretadas como citações, e linhas entre *.R1* e *.R2* 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 em arquivos ps
roff2text	Transforma arquivos roff em arquivos de texto
roff2x	Transforma arquivos roff em outros formatos
soelim	Lê arquivos e substitui linhas da forma *.so arquivo* pelo conteúdo do *arquivo* mencionado
tbl	Compila descrições de tabelas embutidas em arquivos de entrada troff em comandos que são entendidos por **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

8.59. GRUB-2.06

O pacote GRUB contém o GRand Unified Bootloader.

Tempo aproximado de construção: 0,7 UPC
Espaço em disco exigido: 158 MB

8.59.1. Instalação do GRUB

Nota

Se seu sistema tem suporte UEFI e você deseja inicializar LFS com UEFI, então você pode pular esse pacote em LFS, e instalar GRUB com suporte UEFI (e as dependências dele) seguindo *a página BLFS* ao final deste capítulo.

Prepare GRUB para compilação:

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

O significado das novas opções de configure:

`--disable-werror`
Isso permite que a construção complete com avisos introduzidos por mais recentes versões do Flex.

`--disable-efiemu`
Essa opção minimiza o que é construído desabilitando uma característica e 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 limitado ambiente 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
```

Usar GRUB para tornar seu sistema LFS inicializável será discutido em Seção 10.4, “Usando o GRUB para Configurar o Processo de Inicialização”.

8.59.2. 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 se FILE é do tipo especificado
grub-fstest	Ferramenta para depurar o controlador de sistema de arquivos
grub-glue-efi	Processa imagens EFI ia32 e amd64 e cola elas de acordo com formato Apple
grub-install	Instala o GRUB no seu controlador
grub-kbdcomp	Script que converte um esquema xkb em um reconhecido por GRUB
grub-macbless	Bênção estilo Mac sobre arquivos HFS ou HFS+
grub-menulst2cfg	Converte um `menu.lst` do GRUB Legacy em um `grub.cfg` para uso com GRUB 2
grub-mkconfig	Gera um arquivo de configuração grub
grub-mkimage	Faz uma imagem inicializável do GRUB
grub-mklayout	Gera um arquivo de esquema de teclado do GRUB
grub-mknetdir	Prepara um diretório de inicialização de rede GRUB
grub-mkpasswd-pbkdf2	Gera uma senha PBKDF2 encriptada para uso no menu de inicialização
grub-mkrelpath	Faz um caminho de sistema relativo à raiz dele
grub-mkrescue	Faz uma imagem inicializável do GRUB adequada para um disquete ou CDROM/DVD
grub-mkstandalone	Gera uma imagem independente
grub-ofpathname	É um programa auxiliar que imprime o caminho de um dispositivo GRUB
grub-probe	Sonda informação de dispositivo para um caminho ou dispositivo dado
grub-reboot	Configura a entrada de inicialização padrão para o GRUB para a próxima inicialização apenas
grub-render-label	Renderiza .disk_label 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 de inicialização padrão para o GRUB
grub-sparc64-setup	É um programa auxiliar para grub-setup
grub-syslinux2cfg	Transforma um arquivo de configuração syslinux no formato grub.cfg

8.60. Gzip-1.11

O pacote Gzip contém aplicativos para compressão e descompressão de arquivos.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 20 MB

8.60.1. Instalação do Gzip

Prepare Gzip para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.60.2. Conteúdo do Gzip

Aplicativos instalados: gunzip, gzexe, gzip, uncompress (hard link 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 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` em todos os arquivos dados que são arquivos gzipados, de modo que o **gzip** não comprimirá eles novamente; isso pode ser útil quando 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 Recomprime arquivos oriundos do formato **compress** para formato **gzip**—`.Z` para `.gz`

8.61. IPRoute2-5.16.0

O pacote IPRoute2 contém aplicativos para redes baseadas em IPV4 básicas e avançadas.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 15 MB

8.61.1. 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 em LFS. Entretanto, um diretório para **arpd** e uma página de manual ainda serão instalados. Impeça isso executando os comandos abaixo. Se o binário **arpd** for necessário, então instruções para compilar o Berkeley DB podem ser encontradas no Livro BLFS em *https://www.linuxfromscratch.org/blfs/view/11.1/server/db.html*.

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Compile o pacote:

```
make
```

Esse pacote não tem uma suíte de teste funcional.

Instale o pacote:

```
make SBINDIR=/usr/sbin install
```

Se desejado, então instale a documentação:

```
mkdir -pv             /usr/share/doc/iproute2-5.16.0
cp -v COPYING README* /usr/share/doc/iproute2-5.16.0
```

8.61.2. Conteúdo do IPRoute2

Aplicativos instalados: bridge, ctstat (link para lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link para lnstat), ss e tc
Diretórios instalados: /etc/iproute2, /usr/lib/tc e /usr/share/doc/iproute2-5.16.0

Descrições Curtas

bridge Configura pontes de redes

ctstat Utilitário de situação de conexão

genl Frontal utilitário de link de rede genérico

ifcfg Um encapsulador de script de shell para o comando **ip** [Note que ele exige os aplicativos **arping** e **rdisk** originários do pacote iputils encontrado em *http://www.skbuff.net/iputils/*]

ifstat Mostra as estatísticas de interface, incluindo a quantidade de pacotes transmitidos e recebidos pela interface

ip O executável principal. Ele tem várias funções:
ip link *<dispositivo>* permite usuárias(os) olharem para o estado de dispositivos e fazerem mudanças
ip addr permite usuárias(os) olharem para endereços e propriedades deles, adicionarem novos endereços e deletarem antigos

ip neighbor permite usuárias(os) olharem para vínculos de vizinho e propriedades deles, adicionarem novas entradas de vizinho e deletarem as antigas
ip rule permite usuárias(os) olharem para políticas de roteamento e mudar elas
ip route permite usuárias(os) olharem para a tabela de roteamento e mudar regras de tabela de roteamento
ip tunnel permite usuárias(os) olharem para os tuneis IP e propriedades deles, e mudar elas
ip maddr permite usuárias(os) olharem para os endereços multicast e propriedades deles, e mudar elas
ip mroute permite usuárias(os) configurarem, mudarem ou deletarem o roteamento multicast
ip monitor permite usuárias(os) continuamente monitorarem o estado de dispositivos, endereços e rotas

lnstat Fornece estatísticas de rede do Linux; ele é uma substituição difundida e mais completa de características para o antigo aplicativo **rtstat**

nstat Mostra estatísticas de rede

routef Um componente do **ip route**. Isso é para esvaziar as tabelas de roteamento

routel Um componente do **ip route**. Isso é 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 de **ip -o** de volta em um formato legível

rtstat Utilitário de situação de rota

ss Similar ao comando **netstat**; exibe conexões ativas

tc Executável de Controle de Tráfego; isso é para implementações de Quality Of Service (QOS) e Class Of Service (COS)
tc qdisc permite usuárias(os) configurarem a disciplina de enfileiramento
tc class permite usuárias(os) configurarem classes baseadas no agendamento de disciplina de enfileiramento
tc estimator permite usuárias(os) estimarem o fluxo de rede dentro de uma rede
tc filter permite usuárias(os) configurarem a filtragem de pacote QOS/COS
tc policy permite usuárias(os) configurarem as políticas de QOS/COS

8.62. Kbd-2.4.0

O pacote Kbd contém arquivos de tabelas de teclas, fontes de console e utilitários de teclado.

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 33 MB

8.62.1. Instalação do Kbd

O comportamento das teclas backspace e delete não é consistente ao longo dos mapas de teclas no pacote Kbd. A seguinte correção conserta esse problema para mapas de tecla i386:

```
patch -Np1 -i ../kbd-2.4.0-backspace-1.patch
```

Após corrigir, a tecla backspace gera o carácter com código 127 e a tecla delete gera uma sequência bem conhecida de escape.

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, que não está disponível no ambiente chroot.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
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 regular mapa de tecla “by” supõe a codificação ISO-8859-5, e o mapa de tecla CP1251 normalmente é usado. Usuárias(os) de tais idiomas tem que baixar mapas de tecla funcionais separadamente.

Se desejado, então instale a documentação:

```
mkdir -pv           /usr/share/doc/kbd-2.4.0
cp -R -v docs/doc/* /usr/share/doc/kbd-2.4.0
```

8.62.2. Conteúdo do Kbd

Aplicativos instalados: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link para psfxtable), psfgettable (link para psfxtable), psfstriptable (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-2.4.0 e /usr/share/unimaps

Descrições Curtas

chvt Muda o terminal virtual de primeiro plano

deallocvt Desaloca terminais virtuais não usados

dumpkeys Despeja as tabelas de tradução de teclado

fgconsole Imprime o número do terminal virtual ativo

getkeycodes Imprime a tabela de mapeamento de código de escaneamento para código de tecla do kernel

kbdinfo Obtém informação sobre a situação de um console

kbd_mode Relata ou configura o modo de teclado

kbdrate Configura as taxas de repetição e atraso de teclado

loadkeys Carrega as tabelas de tradução de teclado

loadunimap Carrega a tabela de mapeamento unicode para fonte do kernel

mapscrn Um aplicativo obsoleto que costumava carregar uma tabela de mapeamento de caractere de saída definida pela(o) usuária(o) 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

psfstriptable 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 kernel; 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 de teclado

setvtrgb Configura o mapa de cor de console em todos os terminais virtuais

showconsolefont Exibe a fonte de tela de console EGA/VGA atual

showkey Relata os códigos de escaneamento, códigos de tecla e códigos ASCII das teclas pressionadas no teclado

unicode_start Põe o teclado e 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

8.63. Libpipeline-1.5.5

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

Tempo aproximado de construção: 0,1 UPC
Espaço em disco exigido: 9,7 MB

8.63.1. Instalação do Libpipeline

Prepare Libpipeline para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.63.2. Conteúdo do Libpipeline

Biblioteca instalada: libpipeline.so

Descrições Curtas

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

8.64. Make-4.3

O pacote Make contém um aplicativo para controlar a geração de executáveis e outros arquivos não fonte de um pacote a partir de arquivos fonte.

Tempo aproximado de construção: 0,5 UPC
Espaço em disco exigido: 13 MB

8.64.1. Instalação do Make

Prepare Make para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.64.2. Conteúdo do Make

Aplicativo instalado: make

Descrições Curtas

make Automaticamente determina quais partes de um pacote precisam ser (re)compiladas e então emite os comandos relevantes

8.65. Patch-2.7.6

O pacote Patch contém um aplicativo para modificar ou criar arquivos por aplicação de um arquivo “patch” tipicamente criado pelo aplicativo **diff**.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 12 MB

8.65.1. Instalação do Patch

Prepare Patch para compilação:

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.65.2. Conteúdo do Patch

Aplicativo instalado: patch

Descrições Curtas

patch Modifica arquivos de acordo com um arquivo de correção (Um arquivo de correção normalmente é uma listagem de diferenças criada com o aplicativo **diff**. Aplicando essas diferenças aos arquivos originais, **patch** cria as versões corrigidas.)

8.66. Tar-1.34

O pacote Tar fornece a habilidade para criar arquivamentos tar bem como realizar vários outros tipos de manipulação de arquivamento. Tar pode ser usado em arquivamentos previamente criados para extrair arquivos, para armazenar arquivos adicionais, ou para atualizar ou listar arquivos que já foram armazenados.

Tempo aproximado de construção: 1,7 UPC
Espaço em disco exigido: 40 MB

8.66.1. Instalação do Tar

Prepare 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 a(o) usuária(o) `root`, porém como ele está sendo executado em um sistema que foi apenas parcialmente construído, substituir ele está OK.

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Um teste, capabilities: binary store/restore, é conhecido por falhar se ele for executado (LFS carece de selinux), porém será pulado se o kernel do anfitrião não suportar atributos estendidos no sistema de arquivos usado para construir LFS.

Instale o pacote:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.34
```

8.66.2. Conteúdo do Tar

Aplicativo instalado: tar
Diretório instalado: /usr/share/doc/tar-1.34

Descrições Curtas

tar Cria, extrai arquivos originários de, e lista o conteúdo de arquivamentos, também conhecidos como tarballs

8.67. Texinfo-6.8

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

Tempo aproximado de construção: 0,6 UPC
Espaço em disco exigido: 112 MB

8.67.1. Instalação do Texinfo

Prepare Texinfo para compilação:

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

Novamente, conserte um problema ao construir o pacote com Glibc-2.34 ou posterior:

```
sed -e 's/__attribute_nonnull__/__nonnull/' \
    -i gnulib/lib/malloc/dynarray-skeleton.c
```

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

Opcionalmente, instale os componentes pertencentes a uma instalação de TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

O significado do parâmetro de 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 seja instalado posteriormente.

O sistema de documentação Info usa um arquivo de texto simples 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 as 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
```

8.67.2. Conteúdo do Texinfo

Aplicativos instalados: info, install-info, makeinfo (link para texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf e texindex
Bibliotecas instaladas: 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 explanar 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 simples 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

8.68. Vim-8.2.4383

O pacote Vim contém um editor de texto poderoso.

Tempo aproximado de construção: 2,4 UPC
Espaço em disco exigido: 206 MB

Alternativas ao Vim

Se você preferir outro editor—como Emacs, Joe ou Nano—por favor consulte *https://www.linuxfromscratch.org/blfs/view/11.1/postlfs/editors.html* para instruções de instalação sugeridas.

8.68.1. 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 vim para compilação:

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

Compile o pacote:

```
make
```

Para preparar os testes, garanta que usuária(o) `tester` pode escrever na árvore de fonte:

```
chown -Rv tester .
```

Agora execute os testes como usuária(o) `tester`:

```
su tester -c "LANG=en_US.UTF-8 make -j1 test" &> vim-test.log
```

A suíte de teste emite muitos dados binários para a tela. Isso pode causar problemas com as configurações do terminal atual. O problema pode ser evitado redirecionando a saída 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
```

Muitas(os) usuárias(os) estão acostumadas(os) a usar **vi** em vez de **vim**. Para permitir a execução do **vim** quando usuárias(os) 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-8.2.4383`, tornando ela consistente com o local da documentação para outros pacotes:

```
ln -sv ../vim/vim82/doc /usr/share/doc/vim-8.2.4383
```

Se um X Window System vier a ser instalado no sistema LFS, então talvez seja necessário recompilar vim após instalar X. O Vim vem com uma versão GUI do editor que exige o X e algumas bibliotecas adicionais para ser instalada. Para mais informações sobre esse processo, consulte a documentação de vim e a página de instalação de vim no livro BLFS em *https://www.linuxfromscratch.org/blfs/view/11.1/postlfs/vim.html*.

8.68.2. Configurando Vim

Por padrão, **vim** executa em modo incompatível com vi. Isso talvez seja novo para usuárias(os) que usaram 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 àquelas(es) que mudariam para o modo “compatible” que esse deveria ser a primeira configuração no arquivo de configuração. Isso é necessário, pois ela muda outras configurações, e substituições precisam vir após essa configuração. Crie um arquivo de configuração **vim** padrão executando o seguinte:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

" Ensure defaults are set before customizing settings, not after
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

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 **vi** antigo. 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** sobre a 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, vim instala apenas arquivos de soletrar para o idioma inglês. Para instalar arquivos de soletrar para seu idioma preferido, baixe os arquivos `*.spl` e, opcionalmente, o `*.sug` para seu idioma e codificação de caracter a partir de *ftp://ftp.vim.org/pub/vim/runtime/spell/* e salve-os em `/usr/share/vim/vim82/spell/`.

Para usar esses arquivos de soletrar, alguma configuração em `/etc/vimrc` é necessária, por exemplo:

```
set spelllang=en,ru
set spell
```

Para mais informação, veja o arquivo README apropriado localizado na URL acima.

8.68.3. 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 básicas e comandos do **vim**

xxd Cria um despejo hexadecimal do arquivo dado; ele também pode fazer o reverso, de forma que ele pode ser usado para correção de binário

8.69. Eudev-3.2.11

O pacote Eudev contém aplicativos para criação dinâmica de nós de dispositivo.

Tempo aproximado de construção: 0,2 UPC
Espaço em disco exigido: 83 MB

8.69.1. Instalação do Eudev

Prepare Eudev para compilação:

```
./configure --prefix=/usr           \
            --bindir=/usr/sbin      \
            --sysconfdir=/etc       \
            --enable-manpages       \
            --disable-static
```

Compile o pacote:

```
make
```

Crie alguns diretórios agora que são necessários para testes, porém também serão usados como uma parte da instalação:

```
mkdir -pv /usr/lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

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

```
tar -xvf ../udev-lfs-20171102.tar.xz
make -f udev-lfs-20171102/Makefile.lfs install
```

8.69.2. Configurando Eudev

Informação acerca de dispositivos de hardware é mantida nos diretórios `/etc/udev/hwdb.d` e `/usr/lib/udev/hwdb.d`. Eudev precisa que a informação seja compilada em um banco de dados binário `/etc/udev/hwdb.bin`. Crie o banco de dados inicial:

```
udevadm hwdb --update
```

Esse comando precisa ser executado cada vez que a informação de hardware for atualizada.

8.69.3. Conteúdo do Eudev

Aplicativos instalados: udevadm e udevd
Biblioteca instalada: libudev.so
Diretórios instalados: /etc/udev, /usr/lib/udev e /usr/share/doc/udev-udev-lfs-20171102

Descrições Curtas

udevadm Ferramenta de administração udev genérica: controla o daemon udevd, fornece informação a partir do banco de dados do Udev, monitora uevents, aguarda que uevents terminem, testa configuração do Udev e deflagra uevents para um dispositivo dado

udevd Um daemon que ouve uevents no soquete de link de rede, cria dispositivos e executa os aplicativos externos configurados em resposta a esses uevents

`libudev` Uma interface de biblioteca para informação de dispositivo do udev

`/etc/udev` Contém arquivos de configuração do Udev, permissões de dispositivo e regras para nomear dispositivo

8.70. Man-DB-2.10.1

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

Tempo aproximado de construção: 0,3 UPC
Espaço em disco exigido: 39 MB

8.70.1. Instalação do Man-DB

Prepare Man-DB para compilação:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.10.1 \
            --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 de configure:

`--disable-setuid`
Isso desabilita fazer o aplicativo **man** configurar uid para usuária(o) `man`.

`--enable-cache-owner=bin`
Isso torna os arquivos de cache de sistema de propriedade da(o) usuária(o) `bin`.

`--with-...`
Esses três parâmetros são usados para configurar alguns aplicativos padrão. **lynx** é um navegador de rede baseado em texto (veja-se BLFS para instruções de instalação); **vgrind** converte fontes de aplicativo para entrada 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 BLFS, mas você deveria ser capaz de instalá-los após terminar o LFS se você desejar fazer isso.

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

Compile o pacote:

```
make
```

Para testar os resultados, execute:

```
make check
```

Instale o pacote:

```
make install
```

8.70.2. Páginas de Manual não inglesas no LFS

A seguinte tabela mostra o conjunto de caracteres no qual Man-DB supõe que as páginas de manual instaladas sob `/usr/share/man/<ll>` estarão codificadas. Em adição a isto, o Man-DB determina corretamente se páginas de manual instaladas naquele diretório estão codificadas com UTF-8.

Tabela 8.1. Codificação de caracteres esperada das páginas de manual de 8-bit legadas

Idioma (código)	Codificação	Idioma (código)	Codificação
Dinamarquês (da)	ISO-8859-1	Croata (hr)	ISO-8859-2
Alemão (de)	ISO-8859-1	Húngaro (hu)	ISO-8859-2
Inglês (en)	ISO-8859-1	Japonês (ja)	EUC-JP
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	Russo (ru)	KOI8-R
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
Grego (el)	ISO-8859-7		

Nota

Páginas de manual em idiomas que não estão na lista não são suportadas.

8.70.3. 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-2.10.1

Descrições Curtas

accessdb Despeja o conteúdo do banco de dados **whatis** em formato legível por humanos

apropos Pesquisa no banco 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 sobre 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 o banco 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 da(o) usuária(o)

whatis Pesquisa no banco de dados **whatis** e exibe as descrições curtas de comandos do sistema que contém 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**

8.71. Procps-ng-3.3.17

O pacote Procps-ng contém aplicativos para monitorar processos.

Nota

Esse pacote extrai para o diretório `procps-3.3.17`, não o esperado `procps-ng-3.3.17`.

Tempo aproximado de construção: 0,4 UPC
Espaço em disco exigido: 19 MB

8.71.1. Instalação do Procps-ng

Prepare procps-ng para compilação:

```
./configure --prefix=/usr                            \
            --docdir=/usr/share/doc/procps-ng-3.3.17 \
            --disable-static                         \
            --disable-kill
```

O significado da opção de configure:

`--disable-kill`
Essa chave desabilita a construção do comando **kill** que será instalado pelo pacote util-linux.

Compile o pacote:

```
make
```

Para executar a suíte de teste, execute:

```
make check
```

Cinco testes relacionados a pkill são conhecidos por falhar devido a um problema com testes que não foram atualizados.

Instale o pacote:

```
make install
```

8.71.2. Conteúdo do Procps-ng

Aplicativos instalados: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w e watch
Biblioteca instalada: libprocps.so
Diretórios instalados: /usr/include/proc e /usr/share/doc/procps-ng-3.3.17

Descrições Curtas

free Relata a quantidade de memória livre e usada (ambas memória física e swap) no sistema
pgrep Procura por processos baseado nos nomes deles e outros atributos
pidof Relata os PIDs dos aplicativos dados

pkill	Sinaliza processos baseado nos nomes deles e outros atributos
pmap	Relata o mapeamento de memória do processo dado
ps	Lista os processos em execução atualmente
pwait	Aguarda que um processo termine antes de executar.
pwdx	Relata o diretório de trabalho atual de um processo
slabtop	Exibe informações detalhadas de cache de slab do kernel em tempo real
sysctl	Modifica parâmetros do kernel em tempo de execução
tload	Imprime um gráfico da média de carga de sistema atual
top	Exibe uma lista dos processos com maior uso de CPU; ele fornece uma visão contínua da atividade do processador em tempo real
uptime	Relata há quanto tempo o sistema está executando, quantas(os) usuárias(os) estão logadas(os) e as médias de carga de sistema
vmstat	Relata estatísticas de memória virtual, dando informações sobre processos, memória, paginação, Entrada/Saída (E/S) de bloco, traps e atividade da CPU
w	Mostra quais usuárias(os) estão logadas(os) atualmente, onde e desde quando
watch	Executa um comando dado repetidamente, exibindo a primeira tela cheia da saída dele; isso permite que uma(m) usuária(o) observe a mudança de saída ao longo do tempo
`libprocps`	Contém as funções usadas pela maioria dos aplicativos nesse pacote

8.72. Util-linux-2.37.4

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: 1,1 UPC

Espaço em disco exigido: 261 MB

8.72.1. Instalação do Util-linux

Prepare Util-linux para compilação:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime   \
            --bindir=/usr/bin    \
            --libdir=/usr/lib    \
            --sbindir=/usr/sbin  \
            --docdir=/usr/share/doc/util-linux-2.37.4 \
            --disable-chfn-chsh  \
            --disable-login      \
            --disable-nologin    \
            --disable-su         \
            --disable-setpriv    \
            --disable-runuser    \
            --disable-pylibmount \
            --disable-static     \
            --without-python     \
            --without-systemd    \
            --without-systemdsystemunitdir
```

As opções --disable e --without impedem avisos acerca de construir componentes que exigem pacotes ausentes em LFS ou estão inconsistentes com aplicativos instalados por outros pacotes.

Compile o pacote:

```
make
```

Se desejado, execute a suíte de teste como uma(m) usuária(o) não root:

Atenção

Executar a suíte de teste como a(o) usuária(o) `root` pode ser danoso ao seu sistema. Para executá-lo, a opção CONFIG_SCSI_DEBUG para o kernel precisa estar disponível no sistema em execução atualmente e precisa ser construída como um módulo. Construí-lo dentro do kernel impedirá a inicialização. Para cobertura completa, outros pacotes do BLFS precisam ser instalados. Se desejado, esse teste pode ser executado após reiniciar no sistema LFS completo e executar:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

Nota

Existe um teste que falha no ambiente chroot e causa os testes travarem para sempre. O problema não ocorre do lado de fora do ambiente chroot. Para contornar o problema, delete o teste:

```
rm tests/ts/lsns/ioctl_ns
```

```
chown -Rv tester .
su tester -c "make -k check"
```

Instale o pacote:

```
make install
```

8.72.2. 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, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (link para last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lsipc, lsirq, 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, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (link para swapon), swapon, switch_root, taskset, uclampset, ul, umount, uname26, unshare, utmpdump, uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 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-2.37.4 e /var/lib/hwclock

Descrições Curtas

addpart Informa o kernel Linux de 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 Executa comando de zona no dispositivo de bloco dado

blockdev Permite usuárias(os) chamar 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 a pontuação de matador de OOM

chrt Manipula atributos de tempo real de um processo

col	Filtra feeds de linha reversa
colcrt	Filtra saída **nroff** para terminais que não tem algumas capacidades, 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 hard ou soft
delpart	Pede ao kernel Linux para remover uma partição
dmesg	Despeja as mensagens de inicialização do kernel
eject	Ejeta mídia removível
fallocate	Pré-aloca espaço para um arquivo
fdisk	Manipula a tabela de partição do dispositivo dado
fincore	Conta páginas de conteúdo de arquivo em núcleo
findfs	Encontra um sistema de arquivos pelo rótulo ou 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 de controlador de kernel de ioctl de FIFREEZE/FITHAW
fstrim	Descarta blocos não usados em um sistema de arquivos montado
getopt	Analisa opções na linha de comando dada
hexdump	Despeja o arquivo dado em hexadecimal ou em outro formato dado
hwclock	Lê ou configura o relógio de hardware do sistema, também chamado de Real-Time Clock (RTC) ou 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 recurso de Inter-Process Communication (IPC) dado
ipcs	Fornece informação de situação de IPC
irqtop	Exibe informação de contador de interrupção do kernel em visão estilo `top(1)`
isosize	Relata o tamanho de um sistema de arquivos iso9660
kill	Envia sinais para processos
last	Mostra quais usuárias(os) derradeiramente logaram-se (e deslogaram-se), pesquisando de volta ao longo do arquivo `/var/log/wtmp`; ele também mostra inicializações de sistema, desligamentos e mudanças de nível de execução

lastb Exibe as tentativas de login falhas, conforme registrado em `/var/log/btmp`

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 informações sobre todos ou dispositivos de bloco selecionados em um formato semelhante a árvore

lscpu Imprime informação de arquitetura da CPU

lsipc Imprime informação acerca de facilidades de IPC empregadas atualmente no sistema

lsirq Exibe informação de contador de interrupção do kernel

lslocks Lista travas locais de sistema

lslogins Lista informação acerca de contas de usuárias(os), grupos e 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 outras(os) usuárias(os) podem enviar mensagens para o terminal da(o) usuária(o) atual

mkfs Constrói um sistema de arquivos em um dispositivo (geralmente uma partição de disco rígido)

mkfs.bfs Cria um sistema de arquivos Santa Cruz Operations (SCO) bfs

mkfs.cramfs Cria um sistema de arquivos cramfs

mkfs.minix Cria um sistema de arquivos Minix

mkswap Inicializa dispositivo ou arquivo dado para ser usado como uma área de troca

more Um filtro para paginar ao longo de texto uma tela de cada vez

mount Anexa o sistema de arquivos no dispositivo dado a um diretório especificado na árvore do sistema de arquivos

mountpoint Verifica se o diretório é um ponto de montagem

namei Mostra os links simbólicos nos nomes de caminho dados

nsenter Executa um aplicativo com espaços de nome de outros processos

partx Informa ao kernel sobre a 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 kernel

rename Renomeia os arquivos dados, substituindo uma sequência de caracteres dada por outra

renice Altera a prioridade de processos em execução

resizepart Pede ao kernel Linux para redimensionar uma partição

rev	Inverte as linhas de um arquivo dado
rkfill	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 informações de tempo
setarch	Muda a arquitetura relatada em um novo ambiente de aplicativo e configura sinalizadores de personalidade
setsid	Executa o aplicativo dado em uma nova sessão
setterm	Configura atributos do terminal
sfdisk	Um manipulador de tabela de partição de disco
sulogin	Permite `root` se logar; ele normalmente é invocado por **init** quando o sistema entra em modo de usuária(o) única(o)
swaplabel	Permite modificar o UUID e rótulo da área de troca
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	Alterna 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 um processo
ul	Um filtro para traduzir sublinhados em sequências de escape indicando sublinhamento para o terminal em uso
umount	Desconecta um sistema de arquivos da árvore de arquivos do sistema
uname26	Um link simbólico para setarch
unshare	Executa um aplicativo com alguns espaços de nome não compartilhados oriundos do pai
utmpdump	Exibe o conteúdo do arquivo de login dado em um formato mais amigável para a(o) usuária(o)
uuidd	Um daemon usado pela biblioteca UUID para gerar UUIDs baseados em horário em uma forma segura e garantidamente única
uuidgen	Cria novos UUIDs. Cada novo UUID pode razoavelmente ser considerado único entre todos os UUIDs criados, no sistema local e em outros sistemas, no passado e no futuro
uuidparse	Um utilitário para analisar identificadores únicos
wall	Exibe o conteúdo de um arquivo ou, por padrão, a entrada padrão dele, nos terminais de todas(os) as(os) usuárias(os) logadas(os) atualmente
wdctl	Mostra a situação de vigilante de hardware
whereis	Relata o local do binário, fonte e página de manual para o comando dado
wipefs	Limpa uma assinatura de sistema de arquivos a partir de um dispositivo
x86_64	Um link simbólico para setarch
zramctl	Um aplicativo para configurar e controlar dispositivos zram (disco ram comprimido)

`libblkid`	Contém rotinas para identificação de dispositivo e extração de token
`libfdisk`	Contém rotinas para manipular tabelas de partição
`libmount`	Contém rotinas para montagem e desmontagem de dispositivo de bloco
`libsmartcols`	Contém rotinas para auxiliar a saída de tela em forma de tabela
`libuuid`	Contém rotinas para gerar identificadores únicos para objetos que talvez sejam acessíveis além do sistema local

8.73. E2fsprogs-1.46.5

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: 4,4 UPC em um disco giratório, 1,3 UPC em um SSD
Espaço em disco exigido: 93 MB

8.73.1. 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 de configure:

`--enable-elf-shlibs`
Isso cria as bibliotecas compartilhadas as quais alguns aplicativos nesse pacote usam.

`--disable-*`
Isso evita que e2fsprogs construa e instale as bibliotecas `libuuid` e `libblkid`, o daemon `uuidd`, e o encapsulador **fsck**, uma vez que o util-linux instala versões mais recentes.

Compile o pacote:

```
make
```

Para executar os testes, execute:

```
make check
```

Um teste, u_direct_io, é conhecido por falhar em alguns sistemas.

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 do 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 executando 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
```

8.73.2. 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 defeituosos

chattr Muda os atributos de arquivos em um sistema de arquivos `ext2`; ele também muda sistemas de arquivos `ext3`, a versão de registro em diário dos sistemas de arquivos `ext2`

compile_et Um compilador de tabela de erro; ele converte uma tabela de nomes de códigos de erros e mensagens em um arquivo fonte C adequado para uso com a biblioteca `com_err`

debugfs Um depurador de sistema de arquivo; ele pode ser usado para examinar e mudar o estado de um sistema de arquivos `ext2`

dumpe2fs Imprime informação de superblocos e grupo de blocos para o sistema de arquivos presente em um dispositivo dado

e2freefrag Relata informação de fragmentação de espaço livre

e2fsck É usado para verificar, e opcionalmente reparar sistemas de arquivos `ext2` e sistemas de arquivos `ext3`

e2image É usado para salvar dados críticos de sistema de arquivos `ext2` para um arquivo

e2label Exibe ou muda o rótulo de sistema de arquivos no sistema de arquivos `ext2` presente em um dispositivo dado

e2mmpstatus Verifica situação de MMP de um sistema de arquivos ext4

e2scrub Verifica o conteúdo de um sistema de arquivos ext[234] montado

e2scrub_all Verifica todos os sistemas de arquivos ext[234] montados para erros

e2undo Reexecuta o registro de desfazer undo_log para um sistema de arquivos ext2/ext3/ext4 encontrado em um dispositivo [Isso pode ser usado para desfazer uma operação falha por um aplicativo do e2fsprogs]

e4crypt Utilitário de encriptação de sistema de arquivos ext4

e4defrag Desfragmentador online para sistemas de arquivo ext4

filefrag Relatórios sobre quão fragmentado um arquivo específico pode estar

fsck.ext2 Por padrão verifica sistemas de arquivo `ext2` e é um hard link para **e2fsck**

fsck.ext3 Por padrão verifica sistemas de arquivo `ext3` e é um hard link para **e2fsck**

fsck.ext4 Por padrão verifica sistemas de arquivo `ext4` e é um hard link para **e2fsck**

logsave Salva a saída de um comando em um arquivo de registro

lsattr Lista os atributos de arquivos em um segundo sistema de arquivos estendido

mk_cmds Converte uma tabela de nomes de comando e mensagens de ajuda em um arquivo fonte C adequado para uso com a biblioteca de subsistema `libss`

mke2fs Cria um sistema de arquivos `ext2` ou `ext3` no dispositivo dado

mkfs.ext2 Por padrão cria sistemas de arquivos `ext2` e é um hard link para **mke2fs**

mkfs.ext3 Por padrão cria sistemas de arquivos `ext3` e é um hard link para **mke2fs**

mkfs.ext4 Por padrão cria sistemas de arquivos `ext4` e é um hard link para **mke2fs**

mklost+found Usado para criar um diretório `lost+found` em um sistema de arquivos `ext2`; ele pré-aloca blocos de disco para esse diretório para facilitar a tarefa do **e2fsck**

resize2fs Pode ser usado para alargar ou estreitar um sistema de arquivos `ext2`

tune2fs Ajusta parâmetros ajustáveis de sistema de arquivos em um sistema de arquivos `ext2`

`libcom_err` A rotina comum de exibição de erro

`libe2p` Usado por **dumpe2fs**, **chattr** e **lsattr**

`libext2fs` Contém rotinas para habilitar aplicativos de nível de usuária(o) para lidar com um sistema de arquivos `ext2`

`libss` Usado por **debugfs**

8.74. Sysklogd-1.5.1

O pacote sysklogd contém aplicativos para registrar mensagens de sistema, tais como aquelas dadas pelo kernel quando coisas incomuns acontecem.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 0,6 MB

8.74.1. Instalação do Sysklogd

Primeiro, conserte problemas que causam uma falha de segmentação sob certas condições em klogd e conserte uma construção obsoleta de aplicativo:

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make BINDIR=/sbin install
```

8.74.2. Configurando Sysklogd

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

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

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

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

8.74.3. Conteúdo do Sysklogd

Aplicativos instalados: klogd e syslogd

Descrições Curtas

klogd Um daemon de sistema para interceptar e registrar mensagens do kernel

syslogd Registra as mensagens que aplicativos do sistema oferecem para registro [Cada mensagem registrada contém pelo menos uma marca de data e um nome de dispositivo, e normalmente o nome do aplicativo também, mas isso depende do quão confiável o daemon de registro é dito ser]

8.75. Sysvinit-3.01

O pacote Sysvinit contém aplicativos para controlar a inicialização, execução e desligamento do sistema.

Tempo aproximado de construção: menos que 0,1 UPC
Espaço em disco exigido: 1,4 MB

8.75.1. Instalação do Sysvinit

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

```
patch -Np1 -i ../sysvinit-3.01-consolidated-1.patch
```

Compile o pacote:

```
make
```

Esse pacote não vem com uma suíte de teste.

Instale o pacote:

```
make install
```

8.75.2. 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 para um arquivo de registro

fstab-decode Executa um comando com argumentos codificados para fstab

halt Normalmente invoca **shutdown** com a opção *-h*, exceto quando já em nível de execução 0, então ele diz ao kernel para parar o sistema; ele anota no arquivo `/var/log/wtmp` que o sistema está sendo desligado

init O primeiro processo a ser iniciado quando o kernel inicializou o hardware e que 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, de modo que ele não matará o shell pai dele

poweroff Diz ao kernel para parar o sistema e desligar o computador (veja **halt**)

reboot Diz ao kernel para reinicializar o sistema (veja **halt**)

runlevel Relata o nível de execução anterior e o atual, conforme anotado no último registro de nível de execução em `/run/utmp`

shutdown Desliga o sistema de maneira segura, sinalizando todos os processos e notificando todas(os) as(os) usuárias(os) logadas(os)

telinit Diz ao **init** para qual nível de execução mudar

8.76. Acerca dos Símbolos de Depuração

A maioria dos aplicativos e bibliotecas é, por padrão, compilada com símbolos de depuração incluídos (com opção *-g* do **gcc**). Isso significa que quando depurar um aplicativo ou biblioteca que foi compilado com informação de depuração, o depurador pode fornecer não apenas endereços de memória, mas também os nomes das rotinas e variáveis.

Entretanto, a inclusão desses símbolos de depuração alarga um aplicativo ou biblioteca significativamente. O seguinte é um exemplo 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
- Arquivos do Glibc e GCC (`/lib` e `/usr/lib`) com símbolos de depuração: 87 MB
- Arquivos do Glibc e GCC sem símbolos de depuração: 16 MB

Tamanhos talvez variem dependendo de qual compilador e biblioteca C foram usados, mas quando comparar aplicativos com e sem símbolos de depuração, a diferença geralmente será um fator entre dois e cinco.

Como a maioria das(os) usuárias(os) nunca usará um depurador no aplicativo de sistema delas(es), um monte de espaço de disco pode ser recuperado removendo esses símbolos. A próxima seção mostra como remover todos os símbolos de depuração dos aplicativos e bibliotecas.

8.77. Despojando

Esta seção é opcional. Se a(o) pretensa(o) usuária(o) não for uma(m) programadora(r) e não planeja fazer qualquer depuração nos aplicativos do sistema, então o tamanho do sistema pode ser reduzido em cerca de 2 GB removendo os símbolos de depuração de binários e bibliotecas. Isso não causa nenhum inconveniente além de não mais poder depurar os aplicativos completamente.

A maioria das pessoas que usam os comandos mencionados abaixo não experimenta quaisquer dificuldades. Entretanto, é fácil cometer um erro de digitação e tornar o novo sistema inutilizável, de forma que, antes de executar os comandos **strip**, é uma boa ideia produzir uma cópia de segurança do sistema LFS no estado atual dele.

Os símbolos de depuração para bibliotecas selecionadas estão colocados em arquivos separados. Essa informação de depuração é necessária se executar testes de regressão que usam *valgrind* ou *gdb* posteriormente em BLFS.

Observe que **strip** sobrescreverá o arquivo de binário ou biblioteca que ele está processando. Isso pode quebrar os processos usando código ou dados oriundos do arquivo. Se o próprio processo executando o **strip** for afetado, então o binário ou biblioteca sendo despojado pode ser destruído e pode tornar o sistema completamente inutilizável. Para evitar isso, nós copiaremos algumas bibliotecas e binários para `/tmp`, despojaremos elas lá, e instalaremos elas de volta com o comando **install**. Leia a entrada relacionada em Seção 8.2.1, “Problemas de Atualização” para a justificativa para usar o comando **install** aqui.

```
             libstdc++.so.6.0.29
             libitm.so.1.0.0
             libatomic.so.1.2.0"

cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug $LIB $LIB.dbg
    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-2.38.so
               libhistory.so.8.1
               libncursesw.so.6.3
               libm.so.6
               libreadline.so.8.1
               libz.so.1.2.11
               $(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 serão relatados como tendo o formato de arquivo deles não reconhecido. Esses avisos podem ser seguramente ignorados. Eles indicam que aqueles arquivos são scripts em vez de binários.

8.78. Limpando

Finalmente, limpe alguns arquivos extra deixados pela execução de testes:

```
rm -rf /tmp/*
```

Existem também muitos arquivos instalados nos diretórios /usr/lib e /usr/libexec com uma extensão de nome de arquivo de .la. Esses são arquivos "libtool archive". Como já dito, eles somente são úteis quando vincular com bibliotecas estáticas. Eles são desnecessários, e potencialmente danosos, quando se usar bibliotecas compartilhadas dinâmicas, especialmente quando se usar também sistemas de construção não autotools. Para remover eles, execute:

```
find /usr/lib /usr/libexec -name \*.la -delete
```

Para mais informação acerca de arquivos libtool archive, veja a *seção de 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 ele com:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Por fim, remova a conta de usuária(o) 'tester' temporária criada no início do capítulo anterior.

```
userdel -r tester
```

Capítulo 9. Configuração do Sistema

9.1. Introdução

Inicializar um sistema Linux envolve muitas tarefas. O processo precisa montar ambos sistemas de arquivos virtual e real, inicializar dispositivos, ativar a troca, verificar sistemas de arquivos para integridade, montar quaisquer partições ou arquivos de troca, configurar o relógio do sistema, ativar rede, iniciar quaisquer daemons exigidos pelo sistema, e realizar quaisquer outras tarefas personalizadas necessitadas pela(o) usuária(o). Esse processo precisa estar organizado para garantir que as tarefas sejam realizadas na ordem correta, porém, ao mesmo tempo, ser executado o mais rápido possível.

9.1.1. System V

System V é o processo de inicialização clássico que tem sido usado em sistemas Unix e semelhantes a Unix, tais como Linux, desde cerca de 1983. Ele consiste de um aplicativo pequeno, **init**, que configura aplicativos 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 executados pela(o) usuária(o):

```
0 — parar
1 — Modo de usuária(o) única(o)
2 — Multiusuária(o), sem rede
3 — Modo de multiusuária(o) completo
4 — Definível pela(o) usuária(o)
5 — Modo de multiusuária(o) 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

- Talvez 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 kernel até o prompt de login. A conectividade de rede tipicamente é estabelecida cerca de 2 segundos após o 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 de inicialização inteiro.
- Não suporta diretamente características avançadas, como grupos de controle (cgroups), e agendamento de compartilhamento justo por usuária(o).
- Adicionar scripts exige decisões de sequenciamento estático, manuais.

9.2. LFS-Bootscripts-20210608

O pacote LFS-Bootscripts contém um conjunto de scripts para iniciar/parar o sistema LFS na inicialização/desligamento. Os arquivos de configuração e procedimentos 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: 244 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 rede)

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

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

halt Para o sistema

ifdown Para um dispositivo de rede

ifup Inicializa um dispositivo de rede

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

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

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

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

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

rc O script de controle de nível de execução mestre; ele é responsável por executar todos os outros scripts de inicialização, um por um, em uma sequência determinada pelo nome dos links simbólicos sendo processados

reboot Reinicializa o sistema

sendsignals Garante que cada processo seja terminado antes que o sistema reinicialize ou pare

setclock Reconfigura o relógio do kernel para hora local quando o relógio do hardware não está configurado para hora UTC

ipv4-static Fornece a funcionalidade necessária para atribuir um endereço Internet Protocol (IP) estático para uma interface de rede

swap Habilita e desabilita arquivos e partições de troca

sysctl Carrega valores de configuração de sistema a partir do `/etc/sysctl.conf`, se esse arquivo existir, para dentro do kernel em execução

sysklogd Inicia e para os daemons de registro do sistema e kernel

template Um modelo para criar scripts de inicialização personalizados para outros daemons

udev Prepara o diretório `/dev` e inicia o Udev

udev_retry Tenta novamente uevents do udev que falharam, e copia arquivos de regras gerados de `/run/udev` para `/etc/udev/rules.d` se exigido

9.3. Visão Geral do Manuseio de Dispositivos e Módulos

No Capítulo 8, nós instalamos o pacote udev quando eudev foi construído. Antes de entrarmos em detalhes referentes a como isso funciona, um histórico breve dos métodos anteriores de manuseio de dispositivos é oportuno.

Sistemas Linux em geral tradicionalmente usavam um método de criação de dispositivo estático, pelo qual muitos nós de dispositivos eram criados sob `/dev` (as 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 contém um número de chamadas ao aplicativo **mknod** com os números de dispositivo principal e secundário relevantes para cada dispositivo possível que pudesse existir no mundo.

Usando o método udev, somente aqueles dispositivos os quais são detectados pelo kernel obtém nós de dispositivo criados para eles. Como esses nós de dispositivo serão criados a cada vez que o sistema inicializar, eles serão armazenados em um sistema de arquivos `devtmpfs` (um sistema de arquivos virtual 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 2000, um novo sistema de arquivos chamado `devfs` foi mesclado no kernel 2.3.46 e foi feito disponível durante as séries 2.4 de kernels estáveis. Embora ele estivesse presente no próprio fonte do kernel, esse método de criar dispositivos dinamicamente nunca recebeu suporte decisivo das(os) desenvolvedoras(es) do kernel centrais.

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. É aceito geralmente que, se nomes de dispositivo são passíveis de serem configuráveis, então a política de nomenclatura de dispositivo esteja a cargo de uma(m) administradora(r) do sistema, não imposta sobre elas(es) por (quais)qualquer desenvolvedoras(r(es)) específicas(o(s)). O sistema de arquivos `devfs` também sofria com algumas condições que eram inerentes ao projeto dele e não poderiam ser consertadas sem uma revisão substancial do kernel. Ele ficou marcado como obsoleto por um longo período – e foi finalmente removido do kernel em junho de 2006.

Com o desenvolvimento da árvore do kernel 2.5 instável, liberada posteriormente como as séries 2.6 dos kernels estáveis, um novo sistema de arquivos virtual chamado `sysfs` veio a existir. O trabalho do `sysfs` é o de exportar uma visão da configuração de hardware do sistema para processos de espaço de usuária(o). Com essa representação visível ao espaço de usuária(o), a possibilidade de desenvolvimento de um substituto de espaço de usuária(o) para o `devfs` tornou-se muito mais realista.

9.3.2. Implementação do Udev

9.3.2.1. Sysfs

O sistema de arquivos `sysfs` foi mencionado brevemente acima. Alguém talvez questione como o `sysfs` sabe sobre os dispositivos presentes em um sistema e quais números de dispositivo deveriam ser usados para eles. Controladores que tenham sido compilados diretamente no kernel registram os objetos deles com um `sysfs` (devtmpfs internamente) assim que eles são detectados pelo kernel. Para controladores compilados como módulos, esse registro acontecerá quando o módulo for carregado. Assim que o sistema de arquivos `sysfs` for montado (em /sys), os dados os quais os controladores registram com `sysfs` ficam disponíveis para os processos de espaço de usuária(o) e para udevd para processamento (incluindo modificações para nós de dispositivo).

9.3.2.2. Criação de Nó de Dispositivo

Arquivos de dispositivo são criados pelo kernel por meio do sistema de arquivos `devtmpfs`. Qualquer controlador que deseje registrar um nó de dispositivo usará o `devtmpfs` (via o núcleo do controlador) para fazê-lo. Quando uma instância do `devtmpfs` é montada em `/dev`, o nó de dispositivo inicialmente será criado com um nome, permissões e proprietária(o) fixos.

Pouco tempo depois, o kernel enviará um uevent para **udevd**. Baseado nas regras especificadas nos arquivos dentro dos diretórios `/etc/udev/rules.d`, `/usr/lib/udev/rules.d` e `/run/udev/rules.d`, **udevd** criará links simbólicos adicionais para o nó de dispositivo, ou mudará as permissões, proprietária(o), ou grupo deles, ou modificará a entrada de banco de dados do **udevd** interna (nome) para aquele objeto.

As regras nesses três diretórios são numeradas e todos os três diretórios são mesclados. Se **udevd** não puder encontrar uma regra para o dispositivo que ele está 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 talvez tenham apelidos construídos dentro deles. Apelidos são visíveis na saída 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 udev causarão **udevd** 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 correspondem 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 abaixo maneiras pelas quais o carregamento de controladores indesejados pode ser evitado.

O próprio kernel também é capaz de carregar módulos para protocolos de rede, sistemas de arquivos e suporte NLS sob demanda.

9.3.2.4. Lidando com Dispositivos Plugáveis a Quente/Dinâmicos

Quando você conecta um dispositivo, como um tocador de MP3 Universal Serial Bus (USB), o kernel reconhece que o dispositivo agora está conectado e gera um uevent. Esse uevent é então tratado pelo **udevd** 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 kernel não é carregado automaticamente

O Udev só carregará um módulo se ele tiver um apelido específico a 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-5.16.9, 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 udev, execute **modinfo** com o nome de módulo como o argumento. Agora tente localizar o diretório de 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 consertado posteriormente.

Se não existir arquivo `modalias` no diretório relevante sob `/sys/bus`, então isso significa que as(os) desenvolvedoras(es) do kernel ainda não adicionaram suporte modalias para esse tipo de barramento. Com Linux-5.16.9, esse é o caso com barramentos ISA. Espere que esse problema seja consertado em versões do kernel posteriores.

Udev não é planejado para carregar controladores "encapsuladores", tais como *snd-pcm-oss*, e controladores não hardware, tais como *loop*, de maneira alguma.

9.3.3.2. Um módulo do kernel não é carregado automaticamente, e udev não é planejado para carregar ele

Se o módulo "encapsulador" apenas 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 **modprobe** para carregar o encapsulador após o udev carregar o módulo encapsulado. Para fazer isso, adicione uma linha "softdep" ao arquivo `/etc/modprobe.d/<filename>.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 sobre a sintaxe e capacidades "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 de módulo ao arquivo `/etc/sysconfig/modules` em uma linha separada. Isso funciona para módulos encapsuladores também, mas é abaixo do ideal naquele caso.

9.3.3.3. 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. Udev cria um dispositivo incorretamente, ou faz um 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) pelo fornecedor. 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 talvez seja outra manifestação do problema anterior. Se não, e sua regra usar atributos do `sysfs`, então isso talvez seja um problema de temporização do kernel, a ser consertado em kernels posteriores. Por hora, você pode contornar ele criando uma regra que aguarda o atributo do `sysfs` usado e adiciona ele 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. Udev não cria um dispositivo

O texto adicional assume que o driver está construído estaticamente dentro do kernel ou já carregado como um módulo, e que você já verificou que o udev não cria um dispositivo mal nomeado.

Udev não tem informação necessária para criar um nó de dispositivo se um controlador de kernel não exportar os dados dele para o `sysfs`. Isso é mais comum com controladores terceirizados oriundos de fora da árvore do kernel. Crie um nó de dispositivo estático em `/usr/lib/udev/devices` com os números principal/secundário apropriados (veja o arquivo `devices.txt` dentro da documentação do kernel ou a documentação fornecida pela(o) fornecedora(r) de controlador terceirizado). O nó de dispositivo estático será copiado para `/dev` por **udev**.

9.3.3.7. Ordem de nomenclatura de dispositivo muda aleatoriamente após 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 "consertado". Você não deveria confiar que os nomes de dispositivos do kernel sejam estáveis. Em vez disso, crie suas 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 dos vários utilitários *_id instalados pelo udev. Veja-se a Seção 9.4, "Gerenciando Dispositivos" e Seção 9.5, "Configuração de Rede Geral" para exemplos.

9.3.4. Leitura Útil

Documentação útil adicional está disponível nos seguintes sítios:

- A Userspace Implementation of `devfs` *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- The `sysfs` Filesystem *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

9.4. Gerenciando Dispositivos

9.4.1. Dispositivos de Rede

Udev, por padrão, nomeia dispositivos de rede de acordo com dados de Firmware/BIOS ou características físicas como barramento, slot ou endereço MAC. O propósito dessa convenção de nomenclatura é o de garantir que dispositivos de rede sejam nomeados consistentemente e não baseados no horário que a placa de rede foi descoberta. Por exemplo, em um computador que tem duas placas de rede feitas por Intel e Realtek, a placa de rede fabricada pela Intel talvez se torne eth0 e a placa Realtek se torne eth1. Em alguns casos, após uma reinicialização as placas poderiam ser renumeradas de maneira inversa.

No novo esquema de nomenclatura, nomes de dispositivo de rede típicos seriam então 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 Kernel

O esquema de nomenclatura tradicional usando eth0, eth1, etc., pode ser restaurado adicionando-se **`net.ifnames=0`** na linha de comando do kernel. Isso é mais apropriado para aqueles sistemas que tem apenas um dispositivo ethernet do mesmo tipo. Laptops frequentemente tem múltiplas conexões ethernet que são nomeadas eth0 e wlan0 e são também candidatas para esse método. A linha de comando é passada 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 do Udev Personalizadas

O esquema de nomenclatura pode ser personalizado criando-se regras do udev personalizadas. 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 de rede:

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

Nota

Em alguns casos tais como quando endereços MAC foram atribuídos para uma placa de rede manualmente ou em um ambiente virtual como Qemu ou Xen, o arquivo de regras de rede talvez não tenha sido 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 NIC. A primeira linha para cada NIC é uma descrição comentada mostrando os IDs de hardware delas (por exemplo, fornecedor 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 apenas 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 muitas chaves, separadas por vírgulas e espaços em branco opcionais. Essas chaves da regra e uma explanação de cada uma delas estão a seguir:

- `SUBSYSTEM=="net"` - Isso diz a udev para ignorar dispositivos que não sejam placas de rede.
- `ACTION=="add"` - Isso diz a udev para ignorar essa regra para um uevent que não seja um adicionar (uevents "remover" e "mudar" também acontecem, porém não precisam renomear interfaces de rede).
- `DRIVERS=="?*"` - Isso existe de forma que 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 pais delas.
- `ATTR{address}` - O valor dessa chave é o endereço MAC da NIC.
- `ATTR{type}=="1"` - Isso garante que a regra corresponde apenas 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 chave é o nome que 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 de rede antes de prosseguir, e tenha certeza de usar esse valor `NAME` quando criar seus arquivos de configuração abaixo.

9.4.2. Links Simbólicos de CD-ROM

Alguns aplicativos que você talvez queira instalar posteriormente (por exemplo, vários tocadores de mídia) esperam que os links simbólicos `/dev/cdrom` e `/dev/dvd` existam, e apontem para um dispositivo de CD-ROM ou DVD-ROM. Também, talvez seja conveniente colocar referências a esses links simbólicos em `/etc/fstab`. Udev vem com um script que gerará arquivos de regras para criar esses links simbólicos para você, dependendo das capacidades de cada dispositivo, mas você precisa decidir qual de 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 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 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 a usar dependerá de que tipos de mudanças de dispositivo talvez aconteçam. Se você espera o caminho físico para o dispositivo (isto é, as portas e (ou) slots aos quais ele se conecta) 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 talvez morra, e você o substituiria por um dispositivo diferente com as mesmas capacidades e que estaria plugado 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 que aconteça com maior frequência.

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 DVD.

Se você deseja ver os valores que os scripts do udev usarão, então para o dispositivo de CD-ROM apropriado, 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 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 com bind 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 talvez 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 apenas se você precisar dos links simbólicos em ambos os sistemas para apontar 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 ao que você precisa.

9.4.3. Lidando com dispositivos duplicados

Como explicado na Seção 9.3, “Visão Geral do Manuseio de Dispositivos e Módulos”, a ordem na qual dispositivos com a mesma função aparecem em `/dev` é essencialmente aleatória. Por exemplo, se você tem uma câmera web USB e um sintonizador de TV, as vezes `/dev/video0` se refere à câmera e `/dev/video1` se refere ao sintonizador, e as vezes após uma reinicialização a ordem muda. Para todas as classes de hardware, exceto placas de som e placas de rede, isso é consertável criando regras do udev para links simbólicos persistentes personalizados. O caso das placas de rede é abordado separadamente na Seção 9.5, “Configuração de Rede Geral”, e configuração de placa de som pode ser encontrado em *BLFS*.

Para cada um dos seus dispositivos que é provável ter esse problema (mesmo que o problema não exista em sua distribuição Linux atual), encontre o diretório correspondente sob `/sys/class` ou `/sys/block`. Para dispositivos de vídeo, isso talvez seja `/sys/class/video4linux/videoX`. Descubra os atributos que identificam o dispositivo de maneira única (geralmente, IDs de fornecedor 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"

# Persistent symlinks for webcam and tuner
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", SYMLINK+="web
KERNEL=="video*", ATTRS{device}=="0x036f",  ATTRS{vendor}=="0x109e", SYMLINK+="tvt

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 de Rede Geral

9.5.1. Criando Arquivos de Configuração de Interface de Rede

Quais interfaces são levantadas ou derrubadas pelo script de rede usualmente depende dos arquivos em `/etc/sysconfig/`. Esse diretório deveria conter um arquivo para cada interface a ser configurada, tal como `ifconfig.xyz`, onde “xyz” deveria descrever a placa de rede. O nome de interface (por exemplo, eth0) usualmente é apropriado. Dentro desse arquivo estão atributos para essa interface, tais como endereço(s) IP dela, máscaras de subrede, e por aí vai. É necessário que a base do nome do arquivo seja *ifconfig*.

Nota

Se o procedimento na seção anterior não foi usado, udev atribuirá nomes de interface de placa de rede 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** após você ter inicializado seu sistema.

Os nomes de interface dependem da implementação e configuração do daemon udev em execução no sistema. O daemon udev para LFS (instalado na Seção 8.69, “Eudev-3.2.11”) não executará até que o sistema LFS seja inicializado. Assim, não é confiável determinar os nomes de interface sendo usados no sistema LFS executando aqueles comandos na distribuição anfitriã, *mesmo que você esteja no ambiente chroot*.

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

```
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 corresponder com a configuração adequada.

Se a variável ONBOOT estiver configurada para “yes”, então o script de rede 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 script de rede e não será levantada automaticamente. A interface pode ser manualmente iniciada ou parada com os comandos **ifup** e **ifdown**.

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

A variável SERVICE define o método usado para obter o endereço de IP. O pacote LFS-Bootscripts tem um formato de atribuição de IP modular, 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 abordado no livro BLFS.

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

A variável PREFIX contém o número de bits usados na subrede. Cada octeto em um endereço de IP é 8 bits. Se a máscara de rede da subrede for 255.255.255.0, então ela está usando os primeiros três octetos (24 bits) para especificar o número de rede. Se a máscara de rede for 255.255.255.240, então ela estaria usando os primeiros 28 bits. Prefixos mais longos que 24 bits são comumente usados por Internet Service Providers (ISPs) DSL e baseados em cabos. Nesse exemplo (PREFIX=24), a máscara de rede é 255.255.255.0. Ajuste a variável PREFIX de acordo com sua subrede 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 forma 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 administradora(r) de rede, no `/etc/resolv.conf`. Crie o arquivo executando o seguinte:

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

domain <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>

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

A declaração `domain` pode ser omitida ou substituída com uma declaração `search`. Veja-se a página de manual para resolv.conf para mais detalhes.

Substitua *<IP address of the nameserver>* com o endereço de IP do DNS mais apropriado para a configuração. Frequentemente existirá mais que uma entrada (exigências demandam servidores secundários para capacidade de substituto). Se você precisa ou quer apenas um servidor de DNS, então remova a segunda linha *nameserver* do arquivo. O endereço de IP também talvez seja um roteador na rede local.

Nota

Os endereços DNS IPv4 do Google Public são *8.8.8.8* e *8.8.4.4* para IPv4; e *2001:4860:4860::8888* e *2001:4860:4860::8844* para IPv6.

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

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

9.5.4. Personalizando o Arquivo /etc/hosts

Decida acerca do endereço de IP, fully-qualified domain name (FQDN), e possíveis apelidos para uso no arquivo `/etc/hosts`. A sintaxe é:

```
IP_address myhost.example.org aliases
```

A menos que o computador seja para estar visível para a Internet (por exemplo, existe um domínio registrado e um bloco válido de endereços de IP atribuídos—a maioria das(os) usuárias(os) não tem isso), assegure-se de que o endereço de IP está no intervalo de endereço de IP de rede privado. Intervalos válidos são:

```
Private Network Address Range      Normal Prefix
10.0.0.1 - 10.255.255.254           8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

x 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. Um FQDN válido para esse IP poderia ser lfs.example.org.

Mesmo se não se usar uma placa de rede, um FQDN válido ainda é exigido. Isso é necessário para determinados aplicativos operarem corretamente.

Crie o arquivo `/etc/hosts` executando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.1> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

Os valores *<192.168.1.1>*, *<FQDN>* e *<HOSTNAME>* precisam ser mudados para usuárias(os) ou exigências específicas(os) (se atribuído um endereço de IP por uma(m) administradora(r) de rede/sistema e a máquina estará conectada a uma rede existente). O(s) nome(s) de apelido(s) opcional(is) pode(m) ser omitido(s).

A entrada ::1 é o homônimo IPv6 do 127.0.0.1 e representa a interface de loopback IPv6. 127.0.1.1 é uma entrada de loopback reservada especificamente para o FQDN.

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

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

O Linux usa um aparato de inicialização especial chamado SysVinit que é baseado em um conceito de *níveis de execução*. Isso pode ser bem diferente de um sistema para outro, de forma que não pode ser assumido que, porque as coisas funcionam em uma distribuição do Linux em particular, elas deveriam funcionar da mesma forma no LFS também. O LFS tem sua própria maneira de fazer as coisas, mas ele respeita os padrões geralmente aceitos.

O SysVinit (o qual será referido como “init” daqui pra frente) funciona usando um esquema de níveis de execução. Existem sete (numerados de 0 a 6) níveis de execução (atualmente, existem mais níveis de execução, mas eles são para casos especiais e geralmente não são usados. Veja-se `init(8)` para mais detalhes), e cada um deles corresponde às ações que o computador é suposto realizar quando ele inicia. O nível de execução padrão é 3. Aqui estão as descrições dos diferentes níveis de execução conforme eles estão implementados:

0: parar o computador
1: Modo de usuária(o) única(o)
2: Modo de multiusuária(o), sem rede
3: Modo de multiusuária(o), com rede
4: Reservado para personalização, do contrário faz o mesmo que 3
5: Mesmo que 4, ele é usado usualmente para login GUI (como o **xdm** do X ou o **kdm** do KDE)
6: reinicializar o computador

9.6.2. Configurando o Sysvinit

Durante a inicialização do kernel, o primeiro aplicativo que é executado ou é especificado na linha de comando ou, por padrão, **init**. Esse aplicativo lê o arquivo de inicialização `/etc/inittab`. Crie esse arquivo com:

```
cat > /etc/inittab << "EOF"
# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc S

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
EOF
```

Uma explicação desse arquivo de inicialização está na página de manual para *inittab*. Para o LFS, o comando chave que é executado é **rc**. O arquivo de inicialização acima instruirá **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 arquivo de configuração do sistema descritos em seções subsequentes podem ser alternativamente 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 para `/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, uma(m) usuária(o) 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 `rcsysinit.d`, todos contendo um número de links simbólicos. Alguns 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 ele se torna executado. Quando **init** muda 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`*.

Existe uma exceção para essa explicação. Os links que começam com um *S* nos diretórios `rc0.d` e `rc6.d` não farão nada ser iniciado. Eles serão chamados com o parâmetro *`stop`* para parar alguma coisa. A lógica por trás disso é a de que quando uma(m) usuária(o) está para reiniciar ou parar o sistema, nada precisa ser iniciado. O sistema precisa apenas ser parado.

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 **udevd**, aciona quaisquer dispositivos "coldplug" que já tenham sido criados pelo kernel 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 kernel não mais precisa chamar um binário externo. Em vez disso, o **udevd** escutará em um soquete de link de rede os uevents que o kernel gera.

O script de iniciação **/etc/rc.d/init.d/udev_retry** se ocupa de reacionar eventos para subsistemas cujas regras talvez dependam de sistemas de arquivos que não estão montados até que o script **mountfs** seja executado (em particular, `/usr` e `/var` talvez causem isso). Esse script executa após o script **mountfs**, de forma que aquelas regras (se reacionadas) deveriam prosperar na segunda vez. Ele é configurado a partir do 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/rtc.

Para informação acerca de carregamento de módulo de kernel 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 ajustado 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 kernel detecta a capacidade 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 originária do **hwclock** não for a hora local, então as chances são as de que ele esteja configurado para hora UTC. Verifique isso adicionando ou subtraindo a quantidade apropriada de horas para o fuso horário à (da) hora mostrada pelo **hwclock**. Por exemplo, se você estiver atualmente no fuso horário MST, o qual é conhecido também como GMT -0700, então adicione sete horas à hora local.

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"
# Begin /etc/sysconfig/clock

UTC=1

# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=

# End /etc/sysconfig/clock
EOF
```

Uma boa dica que explica 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 talvez 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 kernel 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 saltada. Sem o arquivo de configuração, (ou configurações equivalentes em `rc.site`), o script de inicialização **console** não fará nada.

O script **console** lê o arquivo `/etc/sysconfig/console` para informação de configuração. Decida qual mapa de teclado e fonte de tela serão usados. Vários HOWTOs específicos de idiomas também podem ajudar com isso, veja-se *http://www.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 teclado 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 kernel enviadas para o console como configurado por **dmesg -n**. Níveis válidos são de "1" (sem mensagens) até "8". O nível padrão é "7".

KEYMAP

Essa variável especifica os argumentos para o aplicativo **loadkeys**, tipicamente, o nome do mapa de teclado a carregar, por exemplo, “it”. Se essa variável não estiver configurada, então o script de inicialização não executará o aplicativo **loadkeys**, e o mapa de teclado do kernel padrão será usado. Observe que uns poucos mapas de teclado 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 pai também deveria ser especificado (por exemplo, qwerty/es) para garantir que o mapa de teclado 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 de teclado padrão não for completamente satisfatório e um pequeno ajuste tenha que ser feito. Por exemplo, para incluir o símbolo do Euro em um mapa de teclado 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 de fonte, “-m”, e o nome do mapa de caracteres de aplicação a carregar. Por exemplo, para carregar a fonte “lat1-16” juntamente com o mapa de caracteres de aplicação “8859-1” (já que ele é apropriado nos Estados Unidos da América do Norte), configure essa variável para “lat1-16 -m 8859-1”. Em modo UTF-8, o kernel usa o mapa de caracteres de aplicação para conversão dos códigos de tecla de 8 bits compostos no mapa de teclado para UTF-8, e assim o argumento do parâmetro "-m" deveria ser configurado para a codificação dos códigos de tecla compostos no mapa de teclado.

UNICODE

Configure essa variável para “1”, “yes” ou “true” para colocar o console em modo UTF-8. Isso é útil em locales baseados em UTF-8 e danoso de outra forma.

LEGACY_CHARSET

Para muitos esquemas de teclado, não existe mapa de teclado Unicode padrão no pacote Kbd. O script de inicialização **console** converterá um mapa de teclado disponível para UTF-8 em tempo real se essa variável estiver configurada para a codificação do mapa de teclado não-UTF-8 disponível.

Alguns exemplos:

- Para uma configuração não-Unicode, apenas as variáveis KEYMAP e FONT são geralmente necessárias. Por exemplo, para uma configuração em polonês, alguém usaria:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

- Como mencionado acima, as vezes é necessário ajustar um mapa de teclado padrão um pouco. O exemplo seguinte adiciona o símbolo do Euro ao mapa de teclado alemão:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# End /etc/sysconfig/console
EOF
```

- O seguinte é um exemplo habilitado para Unicode para búlgaro, onde um mapa de teclado UTF-8 padrão existe:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /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"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

- O seguinte exemplo ilustra conversão automática de mapa de teclado de ISO-8859-15 para UTF-8 e habilitação de teclas mortas em modo Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- Alguns mapas de teclado 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: “press Ctrl+. A E para obter Æ” no mapa de teclado padrão). O Linux-5.16.9 interpreta teclas mortas e regras de composição no mapa de teclado corretamente apenas quando os caracteres fonte a serem compostos juntos não são multibyte. Essa deficiência não afeta mapas de teclado 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 “alpha”. A solução é ou evitar o uso de UTF-8, ou instalar o sistema de janelas X que não tem essa limitação no manuseio de entradas dele.

- Para chinês, japonês, coreano e alguns outros idiomas, o console do Linux não pode ser configurado para exibir os caracteres necessários. Usuárias(os) que precisam de tais idiomas deveriam instalar o Sistema de Janelas X, fontes que cobrem os intervalos de caracteres necessários, e o método de entrada adequado (por exemplo, SCIM, suporta uma ampla variedade de idiomas).

Nota

O arquivo `/etc/sysconfig/console` apenas controla a localização do console de texto do Linux. Ele não tem nada a ver com configurar o esquema de teclado adequado 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 de configuração padrão.

9.6.7. Configurando o Script sysklogd

O script `sysklogd` 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@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 seet 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 talvez ou talvez não seja exigido dependendo dos dispositivos presentes no sistema. Se você tiver apenas partições simples e uma placa ethernet, [então] o processo de inicialização provavelmente não precisará esperar por esse comando. Para pular ele, configure a variável OMIT_UDEV_SETTLE=y.
- O script de inicialização `udev_retry` também executa **udev settle** por padrão. Esse comando é necessário por padrão somente se o diretório `/var` for montado separadamente. Isso é porque o relógio precisa do arquivo `/var/lib/hwclock/adjtime`. Outras personalizações talvez 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 do **fsck**, configure a variável VERBOSE_FSCK=y.
- Quando reinicializar, você talvez 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 iniciou (por exemplo agetty), espera um tempo configurado (padrão 3 segundos), e 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 um parâmetro. Por exemplo para remover o atraso em **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 o parâmetro KILLDELAY=0.

9.7. Os Arquivos de Inicialização de Shell do Bash

O aplicativo de shell **/bin/bash** (daqui por diante referenciado como “o shell”) usa uma coleção de arquivos de inicialização para auxiliar a criar um ambiente para executar dentro. Cada arquivo tem um uso específico e talvez afete o login e ambientes interativos diferentemente. Os arquivos no diretório `/etc` fornecem configurações globais. Se um arquivo equivalente existir no diretório home, [então] ele talvez substitua as configurações globais.

Um shell de login interativo é iniciado após um login bem sucedido, usando o **/bin/login**, lendo o arquivo `/etc/passwd`. Um shell de não-login interativo é iniciado na linha de comando (por exemplo, `[prompt]$` **/bin/bash**). Um shell não-interativo está geralmente presente quando um script de shell está executando. Ele é não-interativo porque ele está processando um script e não esperando pela entrada de usuária(o) entre comandos.

Para mais informação, veja-se **info bash** sob a seção *Arquivos de Inicialização do Bash e Shells Interativos*.

Os arquivos `/etc/profile` e `~/.bash_profile` são lidos quando o shell é invocado como um shell de login interativo.

O `/etc/profile` de base abaixo configura algumas variáveis de ambiente necessárias para o suporte ao idioma nativo. Configurá-las adequadamente resulta em:

- A saída dos aplicativos traduzida para o idioma nativo
- 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 locales não ingleses
- A sequência de ordenação alfabética correta para o país
- Tamanho de papel padrão apropriado
- Formatação correta de valores monetário, hora e data

Substitua *<ll>* abaixo com o código de duas letras para o idioma desejado (por exemplo, “en”) e *<CC>* com o código de duas letras para o país apropriado (por exemplo, “GB”). *<charmap>* deveria ser substituído com o mapa de caracteres canônico para seu locale escolhido. Modificadores opcionais, tais como “@euro”, talvez também estejam presentes.

A lista de todos os locales suportados pela Glibc pode ser obtida executando 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 podem 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 um locale particular. Para determinar o nome canônico, execute o seguinte comando, onde *<locale name>* é a saída dada por **locale -a** para seu locale preferido (“en_GB.iso88591” no nosso exemplo).

```
LC_ALL=<locale name> locale charmap
```

Para o locale “en_GB.iso88591”, o comando acima imprimirá:

```
ISO-8859-1
```

Isso resulta em uma configuração de locale final de “en_GB.ISO-8859-1”. É importante que o locale encontrado usando a heurística acima seja testado antes que ele seja adicionado aos arquivos de inicialização do Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

Os comandos acima deveriam imprimir o nome do idioma, a codificação de caracteres usada pelo locale, a moeda local, e o prefixo para discar antes do número de telefone para se alcançar o país. Se quaisquer dos comandos acima falharem com uma mensagem similar àquela mostrada abaixo, [então] isso significa que seu locale ou não foi instalado no Seção 8.5, “Glibc-2.35” ou não é suportado pela instalação padrão da Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Se isso acontecer, [então] você deveria instalar o locale desejado usando o comando **localedef**, ou considere escolher um locale diferente. As instruções posteriores assumem que não existem tais mensagens de erro originárias da Glibc.

Outros pacotes também podem funcionar incorretamente (mas talvez não necessariamente exibam quaisquer mensagens de erro) se o nome do locale não corresponder às expectativas deles. Nesses casos, investigar como outras distribuições do Linux suportam seu locale poderia fornecer alguma informação útil.

Uma vez que as configurações de locale adequadas tenham sido determinadas, crie o arquivo `/etc/profile`:

```
cat > /etc/profile << "EOF"
# Begin /etc/profile

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

Os locales “C” (padrão) e “en_US.utf8” (aquele recomendado para usuárias(os) 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 alto configurado como caracteres inválidos. Esse é o porquê, por exemplo, do comando **ls** substituir eles com pontos de interrogação nesse locale. Também, uma tentativa de enviar correio com tais caracteres a partir do Mutt ou Pine resulta em mensagens de não conformidade com RFC sendo enviadas (o conjunto de caracteres no correio de saída é indicado como “unknown 8-bit”). Então você pode usar o locale “C” apenas se você tiver certeza de que nunca precisará de caracteres de 8 bits.

Locales baseados em UTF-8 não são bem suportados por alguns aplicativos. Trabalho está em progresso para documentar e, se possível, consertar tais problemas, veja-se *https://www.linuxfromscratch.org/blfs/view/11.1/introduction/locale-issues.html*.

9.8. Criando o Arquivo /etc/inputrc

O arquivo `inputrc` é o arquivo de configuração para a biblioteca readline, a qual fornece capacidades de edição enquanto a(o) usuária(o) estiver digitando uma linha a partir do terminal. Ele funciona traduzindo entradas de teclado em ações específicas. Readline é usada pelo bash e maioria dos outros shells, bem como muitos outros aplicativos.

A maioria das pessoas não necessita de funcionalidade específica de usuária(o), de forma que o comando abaixo cria um `/etc/inputrc` global usado por qualquer uma(m) que se logue. Se você mais tarde decidir que você precisa sobrepor os padrões em uma base por usuária(o), [então] você pode criar um arquivo `.inputrc` no diretório home da(o) usuária(o) com os mapeamentos modificados.

Para mais informação sobre como editar o arquivo `inputrc`, veja-se **info bash** sob a seção *Readline Init File*. **info readline** é também uma boa fonte de informação.

Abaixo está um `inputrc` global genérico junto com comentários para explicar o que as várias opções fazem. Observe que comentários não podem estar na mesma linha que comandos. Crie o arquivo usando o seguinte comando:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

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 diretório (/).

Por exemplo, esse arquivo é consultado pelo **chsh** para determinar quando uma usuária desprivilegiada possa mudar o shell de login para a própria conta dela. Se o nome de comando não estiver listado, [então] a usuária terá negada a habilidade de mudar shells.

É uma exigência para aplicativos tais como GDM o qual não publiciza o navegador de face se ele não puder encontrar `/etc/shells`, ou daemons do FTP os quais tradicionalmente proíbem acesso a usuárias(os) com shells não incluídos nesse arquivo.

```
cat > /etc/shells << "EOF"
# Begin /etc/shells

/bin/sh
/bin/bash

# End /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 kernel 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"
# Begin /etc/fstab

# file system  mount-point  type     options             dump  fsck
#                                                              order

/dev/<xxx>     /            <fff>    defaults            1     1
/dev/<yyy>     swap         swap     pri=1               0     0
proc           /proc        proc     nosuid,noexec,nodev 0     0
sysfs          /sys         sysfs    nosuid,noexec,nodev 0     0
devpts         /dev/pts     devpts   gid=5,mode=620      0     0
tmpfs          /run         tmpfs    defaults            0     0
devtmpfs       /dev         devtmpfs mode=0755,nosuid    0     0

# End /etc/fstab
EOF
```

Substitua *<xxx>*; *<yyy>*; e *<fff>* com os valores apropriados para o sistema, por exemplo, `sda2`; `sda5`; e `ext4`. Para detalhes sobre os seis campos nesse arquivo, veja-se **man 5 fstab**.

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 locales 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 kernel o entenda. Isso funciona se a definição de conjunto de caracteres relevante (encontrada sob File systems -> Native Language Support quando da configuração do kernel) tenha sido compilada no kernel 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 arquivo sensível a maiúsculas e minúsculas. Para consertar isso, use a opção especial `utf8` em vez de `iocharset=utf8`, para locales 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 montar controladores flash USB, uma(m) usuária(o) 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 usuárias(os) 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 arquivo insensível a maiúsculas e minúsculas), e a opção `utf8` diz ao kernel 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 de página de código e iocharset padrão para alguns sistemas de arquivos durante a configuração do kernel. 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 há maneira de especificar essas configurações para o sistema de arquivos ntfs em tempo de compilação do kernel.

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` para a entrada apropriada em `/etc/fstab`. Para verificar se o controlador de disco suporta essa opção, execute *hdparm* no controlador de disco aplicável. Por exemplo, se:

```
hdparm -I /dev/sda | grep NCQ
```

retornar uma saída não vazia, [então] a opção é suportada.

Nota: partições baseadas em Logical Volume Management (LVM) não podem usar a opção `barrier`.

10.3. Linux-5.16.9

O pacote Linux contém o kernel do Linux.

Tempo aproximado de construção: 1,5 - 130,0 UPC (tipicamente cerca de 12 UPC)

Espaço em disco exigido: 1200 - 8800 MB (tipicamente cerca de 1700 MB)

10.3.1. Instalação do kernel

Construir o kernel envolve uns poucos passos—configuração; compilação; e instalação. Leia o arquivo `README` na árvore do fonte do kernel para métodos alternativos à maneira que este livro configura o kernel.

Prepare para compilação executando o seguinte comando:

```
make mrproper
```

Isso garante que a árvore do kernel esteja absolutamente limpa. O time do kernel recomenda que esse comando seja executado antes de cada compilação do kernel. Não confie que a árvore do fonte esteja limpa após descompactar.

Existem muitas maneiras de configurar as opções do kernel. Usualmente, isso é feito por meio de uma interface controlada por menu, por exemplo:

```
make menuconfig
```

O significado das variáveis de ambiente do make opcionais:

`LANG=<host_LANG_value> LC_ALL=`

Isso estabelece a configuração do locale para aquela usada no anfitrião. Isso talvez seja necessário para um adequado desenho de linha da interface ncurses do menuconfig em um console de texto linux UTF-8.

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 controlada por menu ncurses. Para outras (gráficas) interfaces, digite **make help**.

Para informação geral sobre configuração do kernel, veja-se *https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt*. O BLFS tem alguma informação relativa a exigências de configuração do kernel particulares de pacotes que estão fora do LFS em *https://www.linuxfromscratch.org/blfs/view/11.1/longindex.html#kernel-config-index*. Informação adicional sobre configurar e construir o kernel pode ser encontrada em *http://www.kroah.com/lkn/*

Nota

Um bom ponto de partida para configurar a configuração do kernel 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 as seguintes características ou o sistema poderia não funcionar corretamente ou inicializar de forma alguma:

```
General setup -->
   < > Enable kernel headers through /sys/kernel/kheaders.tar.xz [CONFIG_IKH
Device Drivers  --->
  Graphics support --->
   Frame buffer Devices --->
      [*] Support for frame buffer devices ----
  Generic Driver Options  --->
   [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
   [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
```

Existem muitas outras opções que talvez sejam desejadas, dependendo das exigências para o sistema. Para uma lista das opções necessárias para pacotes do BLFS, veja-se o *Índice BLFS das Configurações do Kernel* (https://www.linuxfromscratch.org/blfs/view/11.1/longindex.html#kernel-config-index).

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 kernel seguindo *a página do BLFS*.

A justificativa para os itens de configuração acima:

`Enable kernel headers through /sys/kernel/kheaders.tar.xz`
Isso exigirá **cpio** ao se construir o kernel. **cpio** não é instalado por LFS.

`Support for uevent helper`
Ter essa opção configurada talvez interfira com o gerenciamento de dispositivo quando se usar Udev/Eudev.

`Maintain a devtmpfs`
Isso criará nós de dispositivos automatizados os quais são povoados pelo kernel, 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 todas(os) as(os) usuárias(os) do Udev/Eudev.

Alternativamente, **make oldconfig** talvez seja mais apropriado em algumas situações. Veja-se o arquivo `README` para mais informação.

Se desejado, [então] pule a configuração do kernel copiando o arquivo config do kernel, `.config`, a partir do sistema anfitrião (assumindo que ele esteja disponível) para o diretório `linux-5.16.9` desempacotado. Entretanto, nós não recomendamos essa opção. Frequentemente é melhor explorar todos os menus de configuração e criar a configuração do kernel a partir do zero.

Compile a imagem do kernel e módulos:

```
make
```

Se usar módulos do kernel, [então] a configuração do módulo em `/etc/modprobe.d` talvez seja exigida. Informação pertinente à configuração de módulos e kernel está localizada na Seção 9.3, “Visão Geral do Manuseio de Dispositivos e Módulos” e na documentação do kernel no diretório `linux-5.16.9/Documentation`. Também, `modprobe.d(5)` talvez seja de interesse.

A menos que o suporte de módulo tenha sido desabilitado na configuração do kernel, instale os módulos com:

```
make modules_install
```

Depois que a compilação do kernel estiver completa, passos adicionais são exigidos para completar a instalação. Alguns arquivos precisam ser copiados para o diretório `/boot`.

Cuidado

Se o sistema anfitrião tiver uma partição /boot separada, [então] os arquivos copiados abaixo deveriam ir para lá. A maneira mais fácil de fazer isso é vincular /boot no anfitrião (do lado de fora do chroot) à /mnt/lfs/boot antes de prosseguir. Como a(o) usuária(o) root no *sistema anfitrião*:

```
mount --bind /boot /mnt/lfs/boot
```

O caminho para a imagem do kernel talvez 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-5.16.9-lfs-11.1
```

`System.map` é um arquivo de símbolo para o kernel. Ele mapeia os pontos de entrada de função de cada função na API do kernel, assim como os endereços das estruturas de dados do kernel para o kernel em execução. Ele é usado como um recurso quando se investigar problemas do kernel. Emita o seguinte comando para instalar o arquivo de mapa:

```
cp -iv System.map /boot/System.map-5.16.9
```

O arquivo de configuração do kernel `.config` produzido pelo passo **make menuconfig** acima contém todas as seleções de configuração para o kernel que foi recém compilado. É uma boa ideia manter esse arquivo para futura referência:

```
cp -iv .config /boot/config-5.16.9
```

Instale a documentação para o kernel do Linux:

```
install -d /usr/share/doc/linux-5.16.9
cp -r Documentation/* /usr/share/doc/linux-5.16.9
```

É importante observar que os arquivos no diretório do fonte do kernel não são de propriedade da(o) *root*. Sempre que um pacote é desempacotado como a(o) usuária(o) *root* (como nós fizemos dentro do chroot), os arquivos tem os IDs de usuária(o) e grupo do que quer que fossem no computador da(o) empacotadora(r). 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 usuária(o) que a(o) empacotadora(r) usou será atribuído para alguém na máquina. Essa pessoa então teria acesso de escrita ao fonte do kernel.

Nota

Em muitos casos, a configuração do kernel precisará ser atualizada para pacotes que serão instalados posteriormente em BLFS. Diferente de outros pacotes, não é necessário remover a árvore do fonte do kernel depois que o recém construído kernel for instalado.

Se a árvore do fonte do kernel será mantida, [então] execute **chown -R 0:0** no diretório `linux-5.16.9` para assegurar que todos os arquivos são de propriedade da(o) usuária(o) *root*.

Atenção

Alguma documentação do kernel recomenda criar um link simbólico a partir de `/usr/src/linux` apontando para o diretório do fonte do kernel. Isso é específico para kernels 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 talvez deseje construir tão logo seu sistema LFS base 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-5.16.9”. Portanto, eles *nunca* deveriam ser substituídos tanto pelos cabeçalhos do kernel crus quanto por quaisquer outros cabeçalhos sanitizados do kernel.

10.3.2. Configurando a Ordem de Carregamento de Módulos 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 **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 serão carregados na ordem correta; ehci_hcd precisa ser carregado antes de ohci_hcd e uhci_hcd para 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"
# Begin /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# End /etc/modprobe.d/usb.conf
EOF
```

10.3.3. Conteúdo do Linux

Arquivos instalados: config-5.16.9, vmlinuz-5.16.9-lfs-11.1 e System.map-5.16.9
Diretórios instalados: /lib/modules e /usr/share/doc/linux-5.16.9

Descrições Curtas

`config-5.16.9` Contém todas as seleções de configuração para o kernel

`vmlinuz-5.16.9-lfs-11.1`	O motor do sistema Linux. Quando se liga o computador, o kernel 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-5.16.9`	Uma lista de endereços e símbolos; ele mapeia os pontos de entrada e endereços de todas as funções e estruturas de dados no kernel

10.4. Usando o GRUB para Configurar o Processo de Inicialização

Nota

Se seu sistema tiver suporte UEFI e você desejar inicializar o LFS com UEFI, [então] você deveria pular esta página, e configurar o GRUB com suporte UEFI usando as instruções fornecidas na *página do BLFS*.

10.4.1. Introdução

Atenção

Configurar o GRUB incorretamente pode tornar seu sistema inoperável sem um dispositivo de inicialização alternativo, como um CD-ROM ou unidade USB inicializável. Esta seção não é exigida para inicializar seu sistema LFS. Você talvez 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 tem um dispositivo de inicialização, [então] você pode criar um. Para 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. O número da unidade rígida começa do zero, porém o número da partição inicia do um para partições normais e 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 para a 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 da(o) usuária(o) 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, pode 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 kernel linux que você recém construiu na seção anterior) para a nova partição. Você precisará então desmontar a partição e remontar ela como `/boot`. Se você fizer isso, [então] tenha certeza de atualizar `/etc/fstab`.

Usar a 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 ext2
set root=(hd0,2)

menuentry "GNU/Linux, Linux 5.16.9-lfs-11.1" {
        linux   /boot/vmlinuz-5.16.9-lfs-11.1 root=/dev/sda2 ro
}
EOF
```

Nota

A partir da perspectiva do GRUB, os arquivos do kernel são 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.

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 splash gráficas; reprodução de sons; entrada de mouse; etc. Os detalhes dessas opções estão além do escopo desta introdução.

Cuidado

Existe um comando, grub-mkconfig, que pode 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 do Linux comercial, [então] existe uma boa chance de que esse aplicativo será 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.

Talvez 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 por ajuda em algum ponto) descobrir qual versão do LFS está instalada no sistema. Crie esse arquivo executando:

```
echo 11.1 > /etc/lfs-release
```

Dois arquivos descrevendo o sistema instalado talvez sejam usados por pacotes que podem ser instalados no sistema posteriormente, ou em forma de binário ou construindo eles.

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="11.1"
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 de área de trabalho gráficos. Para criar esse arquivo, execute:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="11.1"
ID=lfs
PRETTY_NAME="Linux From Scratch 11.1"
VERSION_CODENAME="<seu nome aqui>"
EOF
```

Tenha certeza de colocar algum tipo de personalização para os campos 'DISTRIB_CODENAME' e 'VERSION_CODENAME' para tornar o sistema unicamente seu.

11.2. Seja Contado

Agora que você terminou o livro, você quer ser contada(o) como uma(m) usuária(o) do LFS? Vá para *https://www.linuxfromscratch.org/cgi-bin/lfscounter.php* e registre-se como uma(m) usuária(o) 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 hora de reinicializar seu computador. Entretanto, você deveria estar ciente de umas poucas coisas. O sistema que você criou neste livro é bastante mínimo, e provavelmente não terá a funcionalidade que você precisaria para ser capaz de seguir em frente. Instalando uns poucos pacotes extras a partir do livro BLFS enquanto ainda em seu ambiente chroot atual, você pode deixar-se em uma posição muito melhor para continuar tão logo você reinicialize em sua nova instalação do LFS. Aqui estão algumas sugestões:

- Um navegador de modo de texto como o *Lynx* permitirá que você facilmente visualize o livro BLFS em um terminal virtual, enquanto constrói pacotes em outro.
- O pacote *make-ca* permitirá que você configure certificados de âncora confiáveis locais, permitindo que o sistema verifique certificados SSL fornecidos por servidores remotos (por exemplo, um sítio da web usando o HTTPS).
- O pacote *GPM* permitirá que você realize ações de copiar/colar em seus terminais virtuais.
- Se você estiver em uma situação onde configuração de IP estático não atende suas exigências de rede, [então] instalar um pacote como *dhcpcd* ou a porção cliente do *dhcp* talvez seja útil.
- Instalar *sudo* talvez seja útil para construir pacotes como uma(m) usuária(o) não root e facilmente instalar os pacotes resultantes em seu novo sistema.
- Se você quiser acessar seu novo sistema a partir de um sistema remoto dentro de um ambiente GUI confortável, [então] instale *openssh*.
- Para tornar a obtenção de arquivos por meio da Internet mais fácil, instale *wget*.
- Para se conectar a um ponto de acesso sem fios para rede, instale *wpa_supplicant*.
- Finalmente, 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 de arquivos virtuais:

```
umount -v $LFS/dev/pts
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/usr
umount -v $LFS/home
umount -v $LFS
```

Desmonte o próprio sistema de arquivos do LFS:

```
umount -v $LFS
```

Agora, reinicialize o sistema com:

```
shutdown -r now
```

Assumindo que o carregador de inicialização GRUB foi configurado como destacado anteriormente, o menu está configurado para inicializar o *LFS 11.1* automaticamente.

Quando a reinicialização estiver completa, o sistema LFS estará pronto para uso e mais software talvez seja adicionado para suprir suas necessidades.

11.4. E agora?

Obrigado por ler este livro LFS. Nós esperamos que você tenha achado este livro útil e tenha aprendido mais sobre o processo de criação do sistema.

Agora que o sistema LFS está instalado, você talvez esteja se perguntando: “E depois?” Para responder a essa pergunta, nós compilamos uma lista de recursos para você.

- Manutenção

  Notificações de defeitos e segurança são relatadas 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 relatórios. Existem vários recursos online que rastreiam tais relatórios, alguns dos quais estão mostrados abaixo:

  - *CERT* (Computer Emergency Response Team)

    O CERT tem uma lista de discussão que publica alertas de segurança a respeito de vários sistemas operacionais e aplicativos. Informação de assinatura está disponível em *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq é uma lista de discussão de segurança de computador de divulgação completa. Ela publica problemas de segurança descobertos recentemente, e ocasionalmente consertos potenciais para eles. Informação de assinatura está disponível em *http://www.securityfocus.com/archive*.

- Beyond Linux From Scratch

O livro Beyond Linux From Scratch cobre procedimentos de instalação para uma ampla gama de software além do escopo do Livro LFS. O projeto BLFS está localizado em *https://www.linuxfromscratch.org/blfs/view/11.1/*.

- LFS Hints

  As Dicas do LFS são uma coleção de documentos educacionais submetidos por voluntários 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 talvez assine se você estiver necessitada(o) de ajuda; quiser se manter atualizada(o) 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/*.

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 second extended file system

ext3 third extended file system

ext4 fourth extended file system

FAQ Frequently Asked Questions

FHS Filesystem Hierarchy Standard

FIFO First-In, First Out

FQDN Fully Qualified Domain Name

FTP File Transfer Protocol

GB 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 user file-creation mask
USB Universal Serial Bus
UTC Coordinated Universal Time

UUID Universally Unique Identifier

VC Virtual Console

VGA Video Graphics Array

VT Virtual Terminal

Apêndice B. Reconhecimentos

Nós gostaríamos de agradecer às seguintes pessoas e organizações por suas contribuições para o Projeto Linux From Scratch.

- *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 BLFS que ajudaram a tornar este livro possível dando suas sugestões; testando o livro; e submetendo relatórios de defeitos; instruções; e suas experiências com a instalação de vários pacotes.

Tradutoras(es)

- *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> – Projeto de tradução do LFS para português
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Projeto de tradução do LFS para alemão
- *Anton Maisak* <info@linuxfromscratch.org.ru> – Projeto de tradução do LFS para russo
- *Elena Shevcova* <helen@linuxfromscratch.org.ru> – Projeto de tradução do LFS para russo

Mantenedoras(es) 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 Phermsawang* <satit@wbac.ac.th> – espelho lfs.phayoune.org
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – espelho lfs.mirror.shizu-net.jp
- *Init World* <http://www.initworld.com/> – espelho lfs.initworld.com

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 McMurchy* <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 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 as vezes até cinco tipos de dependências listadas abaixo. A primeira lista que outros pacotes necessitam estar disponíveis para 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 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. Na maioria dos casos, isso é porque esses pacotes codificarão rigidamente caminhos para binários dentro dos scripts deles. Se não for construído em uma certa ordem, [então] isso poderia resultar em caminhos como /tools/bin/[binário] sendo colocados dentro de scripts instalados para o sistema final. Isso obviamente não é desejável.

A última lista de dependências são pacotes opcionais que não são endereçados no LFS, porém poderiam ser úteis para a(o) usuária(o). Esses pacotes talvez tenham dependências obrigatórias ou opcionais adicionais 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

Instalação depende de: Attr, Bash, Binutils, Coreutils, GCC, Gettext, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução: Attr e Glibc
Suíte de teste depende de: Automake, Diffutils, Findutils e Libtool
Precisa ser instalado antes de: Coreutils, Sed, Tar e Vim
Dependências opcionais: Nenhuma

Attr

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl, Sed e Texinfo
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Automake, Diffutils, Findutils e Libtool
Precisa ser instalado antes de: Acl e Libcap
Dependências opcionais: Nenhuma

Autoconf

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
Suíte de teste depende de: Automake, Diffutils, Findutils, GCC e Libtool
Precisa ser instalado antes de: Automake
Dependências opcionais: *Emacs*

Automake

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
Suíte de teste depende de: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool e Tar
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Bash

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
Suíte de teste depende de: Expect e Shadow
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Xorg*

Bc

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep e Make
Exigido em tempo de execução: Glibc, Ncurses e Readline
Suíte de teste depende de: Gawk
Precisa ser instalado antes de: Linux
Dependências opcionais: Nenhuma

Binutils

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, File, Flex, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo e Zlib
Exigido em tempo de execução: Glibc e Zlib
Suíte de teste depende de: DejaGNU e Expect
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Elfutils*

Bison

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Perl e Sed
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Diffutils, Findutils e Flex
Precisa ser instalado antes de: Kbd e Tar
Dependências opcionais: *Doxygen*

Bzip2

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make e Patch
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: File
Dependências opcionais: Nenhuma

Check

Instalação depende de: Gawk, GCC, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Bash e Gawk
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Coreutils

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Libcap, Make, OpenSSL, Patch, Perl, Sed e Texinfo
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Diffutils, E2fsprogs, Findutils, Shadow e Util-linux
Precisa ser instalado antes de: Bash, Diffutils, Eudev, Findutils e Man-DB
Dependências opcionais: *Expect.pm* e *IO::Tty*

DejaGNU

Instalação depende de: Bash, Coreutils, Diffutils, Expect, GCC, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Expect e Bash
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Diffutils

Instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Perl
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

E2fsprogs

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo e Util-linux
Exigido em tempo de execução: Glibc e Util-linux
Suíte de teste depende de: Procps-ng e Psmisc
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Eudev

Instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Gperf, Make, Sed e Util-linux
Exigido em tempo de execução: Glibc, Kmod, Xz, Util-linux e Zlib
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Expat

Instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make e Sed
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Python e XML::Parser
Dependências opcionais: Nenhuma

Expect

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed e Tcl
Exigido em tempo de execução: Glibc e Tcl
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Tk*

File

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
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *libseccomp*

Findutils

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Bash e Glibc
Suíte de teste depende de: DejaGNU, Diffutils e Expect
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Flex

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
Suíte de teste depende de: Bison e Gawk
Precisa ser instalado antes de: Binutils, IProute2, Kbd, Kmod e Man-DB
Dependências opcionais: Nenhuma

Gawk

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
Suíte de teste depende de: Diffutils
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *libsigsegv*

GCC

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
Suíte de teste depende de: DejaGNU, Expect e Shadow
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *GNAT* e *ISL*

GDBM

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make e Sed
Exigido em tempo de execução: Bash, Glibc e Readline
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Gettext

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
Suíte de teste depende de: Diffutils, Perl e Tcl
Precisa ser instalado antes de: Automake e Bison
Dependências opcionais: Nenhuma

Glibc

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: Nenhuma
Suíte de teste depende de: File
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

GMP

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed e Texinfo
Exigido em tempo de execução: GCC e Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: MPFR e GCC
Dependências opcionais: Nenhuma

Gperf

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc e Make
Exigido em tempo de execução: GCC e Glibc
Suíte de teste depende de: Diffutils e Expect
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Grep

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed e Texinfo
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Gawk
Precisa ser instalado antes de: Man-DB
Dependências opcionais: *PCRE* e *libsigsegv*

Groff

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
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Man-DB e Perl
Dependências opcionais: *ghostscript* e *Uchardet*

GRUB

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
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Gzip

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Bash e Glibc
Suíte de teste depende de: Diffutils e Less
Precisa ser instalado antes de: Man-DB
Dependências opcionais: Nenhuma

Iana-Etc

Instalação depende de: Coreutils
Exigido em tempo de execução: Nenhuma
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Perl
Dependências opcionais: Nenhuma

Inetutils

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
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Tar
Dependências opcionais: Nenhuma

Intltool

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
Suíte de teste depende de: Perl
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

IProute2

Instalação depende de: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, Libcap, Libelf, Cabeçalhos da API do Linux e Zlib
Exigido em tempo de execução: Bash, Coreutils, Glibc, Libcap, Libelf e Zlib
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Berkeley DB* e *iptables*

Jinja2

Instalação depende de: MarkupSafe e Python
Exigido em tempo de execução: MarkupSafe e Python
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Systemd
Dependências opcionais: Nenhuma

Kbd

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
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Kmod

Instalação depende de: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, OpenSSL, Pkg-config, Sed, Xz e Zlib
Exigido em tempo de execução: Glibc, Xz e Zlib
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Eudev
Dependências opcionais: Nenhuma

Less

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses e Sed
Exigido em tempo de execução: Glibc e Ncurses
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Gzip
Dependências opcionais: *PCRE*

Libcap

Instalação depende de: Attr, Bash, Binutils, Coreutils, GCC, Glibc, Perl, Make e Sed
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: IProute2 e Shadow
Dependências opcionais: *Linux-PAM*

Libelf

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc e Make
Exigido em tempo de execução: Glibc e Zlib
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: IProute2 e Linux
Dependências opcionais: Nenhuma

Libffi

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Make e Sed
Exigido em tempo de execução: Glibc
Suíte de teste depende de: DejaGnu
Precisa ser instalado antes de: Python
Dependências opcionais: Nenhuma

Libpipeline

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Check
Precisa ser instalado antes de: Man-DB
Dependências opcionais: Nenhuma

Libtool

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
Suíte de teste depende de: Autoconf, Automake e Findutils
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Linux

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: Nenhuma
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *cpio*

Cabeçalhos da API do Linux

Instalação depende de: Bash, Binutils, Coreutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Perl e Sed
Exigido em tempo de execução: Nenhuma
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

M4

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Bash e Glibc
Suíte de teste depende de: Diffutils
Precisa ser instalado antes de: Autoconf e Bison
Dependências opcionais: *libsigsegv*

Make

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Perl e Procps-ng
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Guile*

Man-DB

Instalação depende de: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed e Xz
Exigido em tempo de execução: Bash, GDBM, Groff, Glibc, Gzip, Less, Libpipeline e Zlib
Suíte de teste depende de: Util-linux
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *libseccomp*

Man-Pages

Instalação depende de: Bash, Coreutils e Make
Exigido em tempo de execução: Nenhuma
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

MarkupSafe

Instalação depende de: Python
Exigido em tempo de execução: Python
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Jinja2
Dependências opcionais: Nenhuma

Meson

Instalação depende de: Ninja e Python
Exigido em tempo de execução: Python
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Systemd
Dependências opcionais: Nenhuma

MPC

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
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: GCC
Dependências opcionais: Nenhuma

MPFR

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed e Texinfo
Exigido em tempo de execução: Glibc e GMP
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Gawk e GCC
Dependências opcionais: Nenhuma

Ncurses

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch e Sed
Exigido em tempo de execução: Glibc
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: Nenhuma

Ninja

Instalação depende de: Binutils, Coreutils, GCC e Python
Exigido em tempo de execução: GCC e Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Meson
Dependências opcionais: *Asciidoc*, *Doxygen*, *Emacs* e *re2c*

OpenSSL

Instalação depende de: Binutils, Coreutils, GCC, Make e Perl
Exigido em tempo de execução: Glibc e Perl
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Coreutils, Kmod e Linux
Dependências opcionais: Nenhuma

Patch

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make e Sed
Exigido em tempo de execução: Glibc e Patch
Suíte de teste depende de: Diffutils
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Ed*

Perl

Instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed e Zlib
Exigido em tempo de execução: GDBM e Glibc
Suíte de teste depende de: Iana-Etc, Less e Procps-ng
Precisa ser instalado antes de: Autoconf
Dependências opcionais: *Berkeley DB*

Pkg-config

Instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt e Sed
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Kmod
Dependências opcionais: *Glib2*

Procps-ng

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Make e Ncurses
Exigido em tempo de execução: Glibc
Suíte de teste depende de: DejaGNU
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Psmisc

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses e Sed
Exigido em tempo de execução: Glibc e Ncurses
Suíte de teste depende de: Nenhuma suíte de teste disponível
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Python

Instalação depende de: Bash, Binutils, Coreutils, Expat, GCC, Gdbm, Gettext, Glibc, Grep, Libffi, Make, Ncurses, OpenSSL, Sed e Util-linux
Exigido em tempo de execução: Bzip2, Expat, Gdbm, Glibc, Libffi, Ncurses, OpenSSL e Zlib
Suíte de teste depende de: GDB e Valgrind
Precisa ser instalado antes de: Ninja
Dependências opcionais: *Berkeley DB*, *libnsl*, *SQLite* e *Tk*

Readline

Instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed e Texinfo

Exigido em tempo de execução: Glibc e Ncurses

Suíte de teste depende de: Nenhuma suíte de teste disponível

Precisa ser instalado antes de: Bash e Gawk

Dependências opcionais: Nenhuma

Sed

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed e Texinfo

Exigido em tempo de execução: Acl, Attr e Glibc

Suíte de teste depende de: Diffutils e Gawk

Precisa ser instalado antes de: E2fsprogs, File, Libtool e Shadow

Dependências opcionais: Nenhuma

Shadow

Instalação depende de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Make e Sed

Exigido em tempo de execução: Glibc

Suíte de teste depende de: Nenhuma suíte de teste disponível

Precisa ser instalado antes de: Coreutils

Dependências opcionais: *CrackLib* e *Linux-PAM*

Sysklogd

Instalação depende de: Binutils, Coreutils, GCC, Glibc, Make e Patch

Exigido em tempo de execução: Glibc

Suíte de teste depende de: Nenhuma suíte de teste disponível

Precisa ser instalado antes de: Nenhuma

Dependências opcionais: Nenhuma

Systemd

Instalação depende de: Acl, Attr, Bash, Binutils, Coreutils, Diffutils, Expat, Gawk, GCC, Glibc, Gperf, Grep, Jinja2, Libcap, Meson, Sed, Util-linux e Zstd

Exigido em tempo de execução: Acl, Attr, Glibc, Libcap e Util-linux

Suíte de teste depende de: Nenhuma

Precisa ser instalado antes de: Nenhuma

Dependências opcionais: *btrfs-progs*, *cURL*, *cryptsetup*, *docbook-xml*, *docbook-xsl-nons*, *elfutils*, *Git*, *gnu-efi*, *GnuTLS*, *iptables*, *kexec-tools*, *libfido2*, *libgcrypt*, *libidn2*, *Libmicrohttpd*, *libpwquality*, *libseccomp*, *libxkbcommon*, *libxslt*, *Linux-PAM*, *lxml*, *LZ4*, *make-ca*, *p11-kit*, *PCRE2*, *Polkit*, *qemu*, *qrencode*, *quota-tools*, *rsync*, *Sphinx*, *tpm2-tss*, *Valgrind* e *zsh*

Sysvinit

Instalação depende de: Binutils, Coreutils, GCC, Glibc, Make e Sed

Exigido em tempo de execução: Glibc

Suíte de teste depende de: Nenhuma suíte de teste disponível

Precisa ser instalado antes de: Nenhuma

Dependências opcionais: Nenhuma

Tar

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

Suíte de teste depende de: Autoconf, Diffutils, Findutils, Gawk e Gzip

Precisa ser instalado antes de: Nenhuma

Dependências opcionais: Nenhuma

Tcl

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make e Sed

Exigido em tempo de execução: Glibc e Zlib

Suíte de teste depende de: Nenhuma

Precisa ser instalado antes de: Nenhuma

Dependências opcionais: Nenhuma

Texinfo

Instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch e Sed
Exigido em tempo de execução: Glibc e Ncurses
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: Nenhuma

Util-linux

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, Eudev, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Libcap, Make, Ncurses, Sed e Zlib
Exigido em tempo de execução: Glibc, Libcap, Ncurses, Readline e Zlib
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Linux-PAM* e *smartmontools*

Vim

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
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Nenhuma
Dependências opcionais: *Xorg*, *GTK+2*, *LessTif*, *Ruby* e *GPM*

XML::Parser

Instalação depende de: Bash, Binutils, Coreutils, Expat, GCC, Glibc, Make e Perl
Exigido em tempo de execução: Expat, Glibc e Perl
Suíte de teste depende de: Perl
Precisa ser instalado antes de: Intltool
Dependências opcionais: Nenhuma

Xz

Instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc e Make
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: Eudev, File, GRUB, Kmod e Man-DB
Dependências opcionais: Nenhuma

Zlib

Instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Make e Sed
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: File, Kmod, Perl e Util-linux
Dependências opcionais: Nenhuma

Zstd

Instalação depende de: Binutils, Coreutils, GCC, Glibc, Gzip, Make e Xz
Exigido em tempo de execução: Glibc
Suíte de teste depende de: Nenhuma
Precisa ser instalado antes de: GCC e Systemd
Dependências opcionais: *LZ4*

Apêndice D. Scripts de inicialização e configuração do sistema versão-20210608

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
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
#############################################################################

. /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."
      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@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

# Attempt to stop all services started by the previous runlevel,
# and killed in this runlevel
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]}
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix
      sysinit_start=/etc/rc.d/rcS.d/S[0-9][0-9]$suffix

      if [ "${runlevel}" != "0" -a "${runlevel}" != "6" ]; then
         if [ ! -f ${prev_start} -a  ! -f ${sysinit_start} ]; then
            MSG="WARNING:\n\n${i} can't be "
            MSG="${MSG}executed because it was not "
            MSG="${MSG}not started in the previous "
            MSG="${MSG}runlevel (${previous})."
            log_warning_msg "$MSG"
            continue
         fi
      fi

      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 functions in this runlevel
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$runlevel.d/S[0-9][0-9]}
      stop=/etc/rc.d/rc$runlevel.d/K[0-9][0-9]$suffix
      prev_start=/etc/rc.d/rc$previous.d/S[0-9][0-9]$suffix

      [ -f ${prev_start} -a ! -f ${stop} ] && continue
   fi

   check_script_status
      if [ "${SCRIPT_STAT}" == "1" ]; then
         SCRIPT_STAT="0"
         continue
      fi

   case ${runlevel} in
      0|6)
         run ${i} stop
         ;;
      *)
         run ${i} start
         ;;
   esac

   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 (1;) to
# the top 256 glyphs of the 9bit font.  This does
# not affect framebuffer consoles

NORMAL="\\033[0;39m"         # Standard console grey
SUCCESS="\\033[1;32m"        # Success is green
WARNING="\\033[1;33m"        # Warnings are yellow
FAILURE="\\033[1;31m"        # Failures are red
INFO="\\033[1;36m"           # Information is light cyan
BRACKET="\\033[1;34m"        # Brackets are blue

# 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

## Screen Dimensions
# Find current screen size
if [ -z "${COLUMNS}" ]; then
   COLUMNS=$(stty size)
   COLUMNS=${COLUMNS##* }
fi

# When using remote connections, such as a serial port, stty size returns 0
if [ "${COLUMNS}" = "0" ]; then
   COLUMNS=80
fi

## Measurements for positioning result messages
COL=$((${COLUMNS} - 8))
WCOL=$((${COL} - 2))

## 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) responsibilty
        # 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_deamon 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) responsibilty
    # 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.

            progname=${program##*/}

            if [[ -e "/run/${progname}.pid" ]]; then
                pidfile="/run/${progname}.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

                # 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/[^/]*$//'`
        progname=`echo "${program}" | sed "s@${prefix}@@"`

        if [ -e "/run/${progname}.pid" ]; then
            rm -f "/run/${progname}.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             #
#         pathname, path to the specified program                            #
#                                                                            #
# 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 progname
    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 "${prefix}" ]; then
           progname="${program}"
        else
           progname=`echo "${program}" | sed "s@${prefix}@@"`
        fi

        # If a PID file exists with that name, assume that is it.
        if [ -e "/run/${progname}.pid" ]; then
            pidfile="/run/${progname}.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 "${pidfile}"`
        # This can optionally be written as 'sed 1q' to repalce 'head -n1'
        # should LFS move /bin/head to /usr/bin/head
    else
        # Use pidof
        pidlist=`pidof "${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 pidfle] {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: $@ - Message                                                                 #
#                                                                                      #
# Return values: Not used                                                              #
########################################################################################
log_success_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SUCCESS_PREFIX}${SET_COL}${SUCCESS_SUFFIX}"

    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: $@ - Message                                                         #
#                                                                              #
# Return values: Not used                                                      #
################################################################################
log_failure_msg()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${FAILURE_PREFIX}${SET_COL}${FAILURE_SUFFIX}"

    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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${WARNING_PREFIX}${SET_COL}${WARNING_SUFFIX}"

    # 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()
```

```
{
    /bin/echo -n -e "${BMPREFIX}${@}"
    /bin/echo -e "${CURS_ZERO}${SKIP_PREFIX}${SET_COL}${SKIP_SUFFIX}"

    # 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()
{
    /bin/echo -n -e "${BMPREFIX}${@}"

    # 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()
{
    /bin/echo -n -e "${@}"

    # 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 failyure 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 : Mount proc, sysfs, and run
#
# 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:            mountvirtfs
# Required-Start:      $first
# Should-Start:
# Required-Stop:
# Should-Stop:
# Default-Start:       S
# Default-Stop:
# Short-Description:   Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# Description:         Mounts /sys and /proc virtual (kernel) filesystems.
#                      Mounts /run (tmpfs) and /dev (devtmpfs).
# 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 /run/shm
      chmod 1777 /run/shm /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

      ln -sfn /run/shm /dev/shm

      (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 "${1}" in
   start)
      # Exit if there's no modules file or there are no
      # valid entries
      [ -r /etc/sysconfig/modules ]             || exit 0
      egrep -qv '^($|#)' /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
#
# Version     : LFS 7.0
#
########################################################################

### 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 stop
      fi

      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /sbin/udevd --daemon

      # Now traverse /sys in order to "coldplug" devices that have
      # already been discovered
      /sbin/udevadm trigger --action=add    --type=subsystems
      /sbin/udevadm trigger --action=add    --type=devices
      /sbin/udevadm trigger --action=change --type=devices

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_SETTLE"; then
         /sbin/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:   Mounts and unmounts swap partitions.
# Description:         Mounts and unmounts 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 filesystmes 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 stop
      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="${msg}were found and have been been "
         msg="${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 stop
      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 devies 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
      ;;

   *)
      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..."

      # As of udev-186, the --run option is no longer valid
      #rundir=$(/sbin/udevadm info --run)
      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 $line ; do
            /sbin/udevadm trigger --subsystem-match=$subsystem --action=add
         done
      done

      # Now wait for udevd to process the uevents we triggered
      if ! is_true "$OMIT_UDEV_RETRY_SETTLE"; then
         /sbin/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 egrep -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

is_true()
{
   [ "$1" = "1" ] || [ "$1" = "yes" ] || [ "$1" = "true" ]
}

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/fb0 ] && use_fb=1 || use_fb=0

    # Figure out the command to set the console into the
    # desired mode
    is_true "${UNICODE}" &&
       MODE_COMMAND="echo -en '\033%G' && kbd_mode -u" ||
       MODE_COMMAND="echo -en '\033%@\033(K' && kbd_mode -a"

    # On framebuffer consoles, font has to be set for each vt in
    # UTF-8 mode. This doesn't hurt in non-UTF-8 mode also.

    ! is_true "${use_fb}" || [ -z "${FONT}" ] ||
       MODE_COMMAND="${MODE_COMMAND} && setfont ${FONT}"

    # Apply that command to all consoles mentioned in
    # /etc/inittab. Important: in the UTF-8 mode this should
    # happen before setfont, otherwise a kernel bug will
    # show up and the unicode map of the font will not be
    # used.

    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 Burgress (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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            $syslog
# Required-Start:      $first localnet
# Should-Start:
# Required-Stop:       $local_fs
# Should-Stop:         sendsignals
# Default-Start:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts kernel and system log daemons.
# Description:         Starts kernel and system log daemons.
#                      /etc/fstab.
# X-LFS-Provided-By:   LFS
### END INIT INFO

# Note: sysklogd is not started in runlevel 2 due to possible
# remote logging configurations
```

```
. /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

      log_info_msg "Starting kernel log daemon..."
      start_daemon /sbin/klogd
      evaluate_retval
      ;;

   stop)
      log_info_msg "Stopping kernel log daemon..."
      killproc /sbin/klogd
      evaluate_retval

      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
      statusproc klogd
      ;;

   *)
      echo "Usage: ${0} {start|stop|reload|restart|status}"
      exit 1
      ;;
esac

exit 0

# End sysklogd
```

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:       3 4 5
# Default-Stop:        0 1 2 6
# Short-Description:   Starts and configures network interfaces.
# Description:         Starts and configures network interfaces.
# X-LFS-Provided-By:   LFS
### END INIT INFO

case "${1}" in
   start)
      # Start all network interfaces
      for file in /etc/sysconfig/ifconfig.*
      do
         interface=${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..."
      killall5 -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..."
      killall5 -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
# Update      : Bruce Dubbs - bdubbs@linuxfromscratch.org
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            reboot
# Required-Start:
# 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
   stop)
      log_info_msg "Restarting system..."
      reboot -d -f -i
      ;;

   *)
      echo "Usage: ${0} {stop}"
      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
#
# Version     : LFS 7.0
#
########################################################################

### BEGIN INIT INFO
# Provides:            halt
# Required-Start:
```

```
# 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
   stop)
      halt -d -f -i -p
      ;;

   *)
      echo "Usage: {stop}"
      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..."
      start_daemon fully_qualified_path
      ;;

   stop)
      log_info_msg "Stopping..."
      killproc fully_qualified_path
      ;;

   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 deliminator 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 deivce
#               pipe creates a pipe, this will ignore the <major> and
#           <minor> fields
#         <major> and <minor> are the major and minor numbers used for
#     the device.
#############################################################################

# End /etc/sysconfig/createfiles
```

D.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 - kpfleming@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
   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 - kpfleming@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 - kpfleming@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
```

```
   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 8.69, “Eudev-3.2.11”.

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"

# Comms devices

KERNEL=="ippp[0-9]*",       GROUP="dialout"
KERNEL=="isdn[0-9]*",       GROUP="dialout"
KERNEL=="isdnctrl[0-9]*",   GROUP="dialout"
KERNEL=="dcbri[0-9]*",      GROUP="dialout"
```

Apêndice F. Licenças do LFS

Este livro [escrito originalmente em inglês] está licenciado sob a licença da Creative Commons Attribution-NonCommercial-ShareAlike 2.0.

A "versão modificada" do livro (traduzida para o idioma português escrito e falado no Brasil) ("Obra derivada") está licenciada sob a Licença de Documentação Livre GNU, versão 1.3 ou qualquer versão posterior publicada pela Free Software Foundation.

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

F.1. Licença da Creative Commons

Creative Commons Legal Code

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. O CREATIVE COMMONS DISPONIBILIZA ESTA INFORMAÇÃO "NO ESTADO EM QUE SE ENCONTRA". O 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 AUTORAL E/OU OUTRAS LEIS APLICÁVEIS. QUALQUER USO DA OBRA QUE NÃO O AUTORIZADO SOB ESTA LICENÇA OU PELA LEGISLAÇÃO AUTORAL É PROIBIDO.

AO EXERCER QUAISQUER DOS DIREITOS À OBRA AQUI CONCEDIDOS, VOCÊ ACEITA E CONCORDA FICAR OBRIGADO 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 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 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 licenciante e indicados no título desta licença: Atribuição, Compartilhamento pela Mesma Licença.

2. Direitos de Uso Legítimo. Nada nesta licença deve ser interpretado de modo a reduzir, limitar ou restringir quaisquer direitos relativos ao uso legítimo, 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. O Licenciante concede a Você uma licença de abrangência mundial, sem royalties, não-exclusiva, perpétua (pela duração do direito autoral aplicável), sujeita aos termos e condições desta Licença, 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. criar e reproduzir Obras Derivadas;

   c. 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. 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 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 por meios digitais a Obra apenas sob os termos desta Licença, e Você deve incluir uma cópia desta licença, ou o Identificador Uniformizado de Recursos (Uniform Resource Identifier) para esta Licença, com cada cópia ou gravação da Obra que Você distribuir, exibir publicamente, executar publicamente, ou executar publicamente por meios digitais. Você não poderá oferecer ou impor quaisquer termos sobre a Obra que alterem ou restrinjam os termos desta Licença ou o exercício dos direitos aqui concedidos aos destinatários. Você não poderá sub-licenciar a Obra. Você deverá manter intactas todas as informações que se referem a esta Licença e à exclusão de garantias. Você não pode distribuir, exibir publicamente, executar publicamente ou executar publicamente por meios digitais a Obra 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 enquanto incorporada em uma Obra Coletiva, mas isto não requer que a Obra Coletiva, à parte da Obra em si, esteja sujeita aos termos desta Licença. Se Você criar uma Obra Coletiva, em havendo notificação de qualquer Licenciante, Você deve, na medida do razoável, remover da Obra Coletiva qualquer referência a este Licenciante ou Autor Original, conforme solicitado. Se você criar uma Obra Derivada, em havendo notificação de qualquer Licenciante, Você deve, na medida do razoável, remover da Obra Derivada qualquer referência a este Licenciante ou ao Autor 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 do 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 aos destinatários, 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 isto 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 através 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 ao Autor Original crédito razoável com relação ao meio ou mídia que Você está utilizando, através da veiculação do nome (ou pseudônimo, se for o caso) do Autor Original, se fornecido; o título da Obra, se fornecido; na medida do razoável, o Identificador Uniformizado de Recursos (URI) que o 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 Original", ou "Roteiro baseado na Obra original de Autor Original"). Tal crédito pode ser implementado de qualquer forma razoável; entretanto, no caso de Obra Derivada ou Obra Coletiva, este crédito aparecerá no mínimo onde qualquer outro crédito comparável de autoria aparece e de modo ao menos tão proeminente quanto este outro crédito de autoria comparável.

e. De modo a tornar claras estas disposições, quando uma Obra for uma composição musical:

   i. Royalties e execução pública. O Licenciante reserva o seu direito exclusivo de coletar, seja individualmente ou através de entidades coletoras de direitos de execução (por exemplo, ECAD, ASCAP, BMI, SESAC), o valor dos seus direitos autorais pela execução pública da obra ou execução pública digital (por exemplo, webcasting) da Obra se esta execução for predominantemente intencionada ou direcionada à obtenção de vantagem comercial ou compensação monetária privada.

   ii. Royalties e Direitos fonomecânicos. O Licenciante reserva o seu direito exclusivo de coletar, seja individualmente ou através de uma entidade designada como seu agente (por exemplo, a agência Harry Fox), royalties relativos a quaisquer gravações que Você criar da Obra (por exemplo, uma versão “cover”) e distribuir, conforme as disposições aplicáveis de direito autoral, se a distribuição feita por Você de versão “cover” for predominantemente intencionada ou direcionada à obtenção de vantagem comercial ou compensação monetária privada.

f. Direitos de Execução Digital pela Internet (Webcasting) e royalties. De modo a evitar dúvidas, quando a Obra for uma gravação de som, o Licenciante reserva o seu direito exclusivo de coletar, seja individualmente ou através de entidades coletoras de direitos de execução (por exemplo, SoundExchange ou ECAD), royalties e direitos autorais pela execução digital pública (por exemplo, Webcasting) da Obra, conforme as disposições aplicáveis de direito autoral, se a execução digital pública feita por Você for predominantemente intencionada ou direcionada à obtenção de vantagem comercial ou compensação monetária privada.

5. Declarações, Garantias e Exoneração

   EXCETO QUANDO FOR DE OUTRA FORMA MUTUAMENTE ACORDADO PELAS PARTES POR ESCRITO, O 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, ESTAS EXCLUSÕES PODEM NÃO SE APLICAR A VOCÊ.

6. Limitação de Responsabilidade. EXCETO NA EXTENSÃO EXIGIDA PELA LEI APLICÁVEL, EM NENHUMA CIRCUNSTÂNCIA O 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 LICENCIANTE TENHA SIDO AVISADO 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 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 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 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, isto 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 feito por escrito e assinado 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 Licenciante não será obrigado 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 Licenciante e Você.

Importante

O Creative Commons não é uma parte desta Licença e não presta qualquer garantia relacionada à Obra. O 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 o Creative Commons tiver expressamente se identificado como o Licenciante, ele deverá ter todos os direitos e obrigações do 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 ao Creative Commons sem consentimento prévio e por escrito do Creative Commons. Qualquer uso permitido deverá ser de acordo com as diretrizes do Creative Commons de utilização da marca então válidas, conforme sejam publicadas em seu website ou de outro modo disponibilizadas periodicamente mediante solicitação.

O Creative Commons pode ser contactado pelo endereço: *http://creativecommons.org/*.

F.2. A Licença do MIT

Direitos autorais © 1999-2022 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 é 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.

F.3. A Licença de Documentação Livre GNU

Direitos autorais © 1999-2022 Jamenson Espindula

Versão 1.3, 03 de novembro de 2008

Direitos autorais © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. http://fsf.org/

A qualquer pessoa é permitido copiar e distribuir cópias literais deste documento de licença, porém modificá-lo não é permitido.

1. PREÂMBULO

   O propósito desta licença é tornar um manual, livro de texto, ou outro documento funcional e útil livre no sentido da liberdade: para assegurar a qualquer pessoa a liberdade efetiva para copiar e redistribuí-lo, com ou sem modificações, ambos comercialmente ou não comercialmente. Secundariamente, esta Licença preserva para o autor e editor uma maneira de obter crédito pelos seus trabalhos, ao mesmo tempo não sendo considerado responsável por modificações feitas por outros. Esta Licença é uma espécie de “copyleft” (“esquerdos autorais”), o que significa que trabalhos derivados do documento devem necessariamente eles mesmos serem livres no mesmo sentido. Ela complementa a Licença Pública Geral GNU, a qual é uma licença de esquerdos autorais projetada para software livre. Nós projetamos esta Licença para utilizá-la para manuais para software livre, porque software livre precisa de documentação livre: um programa livre deveria vir com manuais provendo as mesmas liberdades que o software provê. Porém esta Licença não é limitada a manuais de software; ela pode ser utilizada para qualquer trabalho textual, independentemente de questões de assunto ou se o trabalho textual for publicado como um livro impresso. Nós recomendamos esta Licença principalmente para trabalhos cujo propósito seja instrução ou referência.

2. APLICABILIDADE E DEFINIÇÕES

   Esta Licença se aplica a qualquer manual ou outro trabalho, em qualquer meio, que contenha um aviso colocado pelo detentor dos direitos autorais dizendo que ele pode ser distribuído sob os termos desta Licença. Tal aviso concede uma licença mundial, livre de patente, ilimitada na duração, para utilizar aquele trabalho sob as condições nela declaradas. O “Documento”, abaixo, se refere a quaisquer desses manuais ou trabalhos. Qualquer membro do publico é um titular da licença, e é mencionado como “você”. Você aceita a licença se você copiar, modificar ou distribuir o trabalho em uma forma que exija permissão sob lei de direitos autorais. Uma “Versão Modificada” do Documento significa qualquer trabalho contendo o Documento ou uma porção dele, seja literalmente copiado, ou com modificações e/ou traduzido em outra língua. Uma “Seção Secundária” é um apêndice nomeado ou uma seção pré-textual do Documento que lida exclusivamente com o relacionamento dos editores ou autores do Documento para com o assunto global do Documento (ou com questões relacionadas) e não contém nada que possa se conformar diretamente com aquele assunto global. (Assim, se o Documento for em parte um livro texto de matemática, uma Seção Secundária não pode explanar nada acerca de cálculos matemáticos). O relacionamento poderia ser uma questão de conexão histórica com o assunto ou com questões relacionadas, ou de posicionamento legal, comercial, filosófico, ético ou político respeitante a eles. As “Seções Invariantes” são certas Seções Secundárias cujos títulos são projetados, como sendo aqueles de Seções Invariantes, no aviso que diz que o Documento é publicado sob esta Licença. Se uma seção não se encaixa na definição de Secundária acima, então a seção não está autorizada a ser designada como Invariante. O Documento pode conter zero Seções Invariantes. Se o Documento não identifica quaisquer Seções Invariantes, então não existe nenhuma. Os “Textos de Capa” são certas passagens curtas de texto que são listadas, como Textos de Primeira Capa ou Textos de Quarta-Capa, no aviso que diz que o Documento é publicado sob esta Licença. Um Texto de Primeira Capa pode ter no máximo cinco (05) palavras, e um Texto de Quarta Capa pode ter no máximo vinte e cinco (25) palavras. Uma cópia “Transparente” do Documento significa uma cópia legível por máquina, representada em um formato cuja especificação está disponível para o público em geral, que é adequada para revisar o documento diretamente com editores de texto genéricos ou (para imagens compostas de pixeis) programas de pintura genéricos ou (para desenhos) algum editor de desenho disponível amplamente, e que seja adequado para entrada a formatadores de texto ou para tradução automática a uma variedade de formatos próprios para entrada a formatadores de texto. Uma cópia feita em um formato de arquivo contrário ao Transparente, cuja linguagem de marcação, ou ausência de linguagem de marcação, tenha sido organizada para frustrar ou desencorajar modificações subsequentes por leitores, não é Transparente. Um formato

de imagem não é Transparente se utilizado para qualquer quantidade substancial de texto. Uma cópia que não é “Transparente” é chamada “Opaca”. Exemplos de formatos adequados para cópias Transparentes incluem ASCII puro sem marcações; formato de entrada Texinfo; formato de entrada LaTeX; SGML ou XML utilizando um DTD disponível publicamente; HTML simples conformante com o padrão; PostScript ou PDF projetado para modificação humana. Exemplos de formatos transparantes de imagens incluem PNG, XCF e JPG. Formatos opacos incluem formatos proprietários que podem ser lidos e editados somente por processadores proprietários de palavra; SGML ou XML para os quais o DTD e/ou as ferramentas de processamentos não estejam disponíveis genericamente; e o HTML gerado por máquina; PostScript ou PDF produzidos por alguns processadores de palavra apenas para propósitos de saída. A “Página de Título” significa, para um livro impresso, a própria página de título, mais tantas páginas seguintes quantas sejam necessárias para manter, legivelmente, o material que esta Licença exige para aparecer na página de título. Para trabalhos em formatos que não tenham qualquer página de título como tal, “Página de Título” significa o texto próximo da mais proeminente aparição do título do trabalho, precedendo o início do corpo do texto. O “editor” significa qualquer pessoa ou entidade que distribui cópias do Documento ao público. Uma seção “Intitulada XYZ” significa uma subunidade nomeada do Documento cujo título ou é precisamente XYZ ou contém XYZ entre parênteses seguinte ao texto que traduz XYZ em outra linguagem. (Aqui XYZ significa um nome específico de seção mencionado abaixo, tais como “Agradecimentos”; “Dedicatórias”; “Patrocínios”; ou “Histórico”). “Preservar o Título” de tal seção quando você modificar o Documento significa que ele permanece uma seção “Intitulada XYZ” de acordo com essa definição. O Documento pode incluir Declarações de Garantia próximas ao aviso que declara que esta Licença se aplica ao Documento. Essas Declarações de Garantia são consideradas como inclusas por referência nesta Licença, porém somente com relação à negação de garantias: qualquer outra implicação que essas Declarações de Garantia possam ter é inválida e não tem efeito sobre o significado desta Licença.

3. CÓPIA LITERAL

   Você pode copiar e distribuir o Documento em qualquer meio, ambos comercialmente e não comercialmente, contanto que esta Licença, os avisos de direitos autorais, e o aviso de licença dizendo que esta Licença se aplica ao Documento estejam reproduzidas em todas as cópias, e que você não adiciona quaisquer outras condições, quaisquer que sejam, àquelas desta Licença. Você não pode utilizar medidas técnicas para obstruir ou controlar a leitura ou posteriores cópias das cópias que você fizer ou distribuir. Entretanto, você pode aceitar remuneração em troca das cópias. Se você distribui um número de cópias grande o suficiente, você deve necessariamente também seguir as condições na seção três (3). Você também pode ceder cópias, sob as mesmas condições declaradas acima, e você pode publicamente exibir cópias.

4. CÓPIAS EM QUANTIDADE

   Se você publicar cópias impressas (ou cópias em mídia que geralmente tem capas impressas) do Documento, em número maior que cem (100), e o aviso de licença do Documento exigir Textos de Capa, você deve necessariamente encartar as cópias em capas que transportem, claramente e legivelmente, todos estes Textos de Capa: Textos de Primeira Capa na primeira capa, e Textos de Quarta Capa na capa traseira. Ambas as capas devem necessariamente também claramente e legivelmente identificar você como o editor dessas cópias. A capa frontal deve necessariamente apresentar o título completo com todas as palavras do título igualmente proeminentes e visíveis. Você pode adicionar outros materiais nas capas adicionalmente. As cópias com modificações limitadas às capas, tanto quanto preservem o título do Documento e satisfaçam essas condições, podem ser tratadas como cópias literais em relação a outros aspectos. Se os textos exigidos para ambas as capas forem muito volumosos para caber legivelmente, você deveria colocar os primeiros listados (tantos quantos caibam razoavelmente) na capa atual, e continuar o restante em páginas adjacentes. Se você publicar ou distribuir cópias Opacas do Documento em número maior que cem (100), você deve necessariamente ou incluir uma cópia Transparente, legível por máquina, junto com cada cópia Opaca, ou declarar, na ou com cada cópia Opaca, uma localização de rede de computador, a partir da qual o público usuário de rede geral tenha acesso para baixar, utilizando protocolos de rede de padrão público, uma

cópia Transparente completa do Documento, livre do material adicionado. Se você se utilizar da última opção, você deve necessariamente adotar razoavelmente passos prudentes, quando você iniciar a distribuição de cópias Opacas em quantidade, para se assegurar que essa cópia Transparente permanecerá então acessível na localização declarada até pelo menos um ano após a última vez que você distribuiu uma cópia Opaca (diretamente ou por intermédio dos seus agentes ou varejistas) daquela edição ao público. É pedido, mas não exigido, que você contate os autores do Documento bem antes de redistribuir qualquer número grande de cópias, para dá-los a oportunidade de lhe fornecer uma versão atualizada do Documento.

5. MODIFICAÇÕES

Você pode copiar e distribuir uma Versão Modificada do Documento sob as condições das seções dois (2) e três (3) acima, contanto que você publique a Versão Modificada precisamente sob esta Licença, com a Versão Modificada preenchendo a função do Documento, portanto licenciando a distribuição e modificação da Versão Modificada a quem quer que possua uma cópia dela. Adicionalmente, você deve necessariamente fazer estas coisas na Versão Modificada:

a. Utilize na Página de Título (e nas capas, se existentes) um título distinto daquele do Documento, e daqueles das versões prévias (as quais deveriam, se existiu alguma, serem listadas na seção Histórico do Documento). Você pode utilizar o mesmo título que uma versão prévia, se o editor original daquela versão conceder permissão.

b. Liste na Página de Título, como autores, uma ou mais pessoas ou entidades responsáveis pela autoria das modificações na Versão Modificada, junto com ao menos cinco dos autores principais do Documento (todos os autores principais, se tiver menos que cinco), a menos que eles liberem você dessa exigência.

c. Declare na Página de Título o nome do editor da Versão Modificada, como o editor.

d. Preserve todos os avisos de direitos autorais do Documento.

e. Adicione um aviso apropriado de direitos autorais para suas modificações, adjacente aos outros avisos de direitos autorais.

f. Inclua, imediatamente após os avisos de direitos autorais, um aviso de licença concedendo ao público permissão para utilizar a Versão Modificada sob os termos desta Licença, na forma mostrada no Adendo abaixo.

g. Preserve, naquele aviso de licença, as listas completas de Seções Invariantes e Textos de Capa exigidos dados no aviso de licença do Documento.

h. Inclua uma cópia inalterada desta Licença.

i. Preserve a seção intitulada “Histórico”, Preserve seu Título, e adicione a ele um item declarando ao menos o título, ano, novos autores, e editor da Versão Modificada, conforme dado na Página de Título. Se não existir uma seção intitulada “Histórico” no Documento, crie uma declarando o título, ano, autores, e editor do Documento, conforme dado em sua Página de Título, então adicione um item descrevendo a Versão Modificada, conforme declarado na frase prévia.

j. Preserve a localização de rede, se existente, dada no Documento para acesso público a uma cópia Transparente do Documento, e da mesma forma as localizações de rede dadas no Documento para versões prévias nas quais foi baseado. Essas podem ser colocadas na seção “Histórico”. Você pode omitir uma localização de rede para um trabalho que foi publicado nos últimos quatro anos anteriores à publicação do próprio do Documento, ou se o editor original da versão à qual a localização de rede se refere conceder permissão.

k. Para cada seção Intitulada “Agradecimentos” ou “Dedicatórias”, Preserve o Título da seção, e preserve na seção toda a substância e tonalidade de cada um dos agradecimentos a contribuidores e/ou dedicatórias dadas nela.

l. Preserve todas as Seções Invariantes do Documento, inalteradas em seus textos e em seus títulos. Os números de Seção ou o equivalente não são considerados parte dos títulos de seção.

m. Delete quaisquer seções Intituladas “Patrocínios”. Tal seção não pode ser incluída na Versão Modificada.

n. Não reintitule qualquer seção existente para Intitulada “Patrocínios” ou para conflitar no título com qualquer Seção Invariante.

o. Preserve quaisquer Declarações de Garantia.

Se a Versão Modificada incluir novas seções pré textuais ou apêndices que se qualifiquem como Seções Secundárias e não contenham material copiado a partir do Documento, você pode, a sua escolha, designar algumas ou todas essas seções como Invariantes. Para fazer isso, adicione seus títulos à lista das Seções Invariantes no aviso de licença da Versão Modificada. Esses títulos devem necessariamente serem distintos de quaisquer outros títulos de seções. Você pode adicionar uma seção Intitulada “Patrocínios”, contanto que ela não contenha nada além de patrocínios da sua Versão Modificada por vários patrocinadores—por exemplo, declarações de avaliadores ou aquelas de que o texto foi aprovado por uma organização como a definição autorizativa de um padrão. Você pode adicionar uma passagem de até cinco palavras, como um Texto de Primeira Capa, e uma passagem de até vinte e cinco palavras, como um Texto de Quarta Capa, ao final da lista dos Textos de Capa na Versão Modificada. Somente uma passagem de Texto de Primeira Capa e uma de Texto de Quarta Capa podem ser adicionadas por (ou mediante acordos feitos por) qualquer uma entidade. Se o Documento já inclui um texto de capa para a mesma capa, previamente adicionado por você ou por acordo feito pela mesma entidade pela qual você está atuando, você não pode adicionar outro; porém você pode substituir o antigo, na permissão explícita do editor prévio que adicionou o antigo. O(s) autor(s) e editor(s) do Documento, por esta Licença, não concedem permissão para utilizar seus nomes para publicidade para ou para afirmar ou implicar patrocínio de qualquer Versão Modificada.

6. COMBINANDO DOCUMENTOS

Você pode combinar o Documento com outros documentos publicados sob esta Licença, sob os termos definidos na seção quatro (4) acima para versões modificadas, contanto que você inclua na combinação todas as Seções Invariantes de todos os documentos originais, não modificados, e listá-los todos como Seções Invariantes do seu trabalho combinado no seu aviso de licença, e você preserva todas as Declarações de Garantias deles. O trabalho combinado precisa conter somente uma cópia desta Licença, e múltiplas Seções Invariantes idênticas podem ser substituídas por uma cópia única. Se existirem múltiplas Seções Invariantes com o mesmo nome, mas conteúdos diferentes, torne o título de cada uma de tal seção único adicionando ao final dele, entre parênteses, o nome do autor ou editor original daquela seção se conhecido, ou, do contrário, um número único. Faça o mesmo ajuste aos títulos da seção na lista de Seções Invariantes no aviso de licença do trabalho combinado. Na combinação, você deve necessariamente combinar quaisquer seções Intituladas “Histórico” nos vários documentos originais, formando uma seção Intitulada “Histórico”; de mesma maneira, combine quaisquer seções Intituladas “Agradecimentos”, e quaisquer seções Intituladas “Dedicatórias”. Você deve necessariamente deletar todas as seções Intituladas “Patrocínios”.

7. COLEÇÕES DE DOCUMENTOS

Você pode produzir uma coleção consistente do Documento e outros documentos publicados sob esta Licença, e substitua as cópias individuais desta Licença nos vários documentos por uma cópia única que esteja incluída na coleção, contanto que você siga as regras desta Licença para cópias literais de cada um dos documentos em todos os outros aspectos. Você pode extrair um documento único de tal coleção, e distribuí-lo individualmente sob esta Licença, contanto que você insira uma cópia desta Licença no documento extraído, e siga esta Licença em todos os outros aspectos relativos à cópias literais daquele documento.

8. AGREGAÇÃO COM TRABALHOS INDEPENDENTES

Uma compilação do Documento ou seus derivados com outros documentos separados e independentes ou trabalhos, dentro ou junto a volume de armazenamento ou meio de distribuição, é chamado em “agregado” se os direitos autorais resultantes da compilação não forem utilizados para limitar os direitos legais dos usuários da compilação

além do que os trabalhos individuais permitem. Quando o Documento for incluído em um agregado, esta Licença não se aplica aos outros trabalhos no agregado, os quais não são eles próprios trabalhos derivados do Documento. Se a exigência do Texto de Capa da seção três (3) for aplicável a essas cópias do Documento, então se o Documento for menor que a metade do agregado inteiro, os Textos de Capa do Documento podem ser colocados em capas que encartem o Documento dentro do agregado, ou o equivalente eletrônico de capas se o Documento estiver em formato eletrônico. Do contrário, eles devem necessariamente aparecer nas capas impressas que encartem o agregado inteiro.

9. TRADUÇÃO

Tradução é considerada um tipo de modificação, de forma que você pode distribuir traduções do Documento sob os termos da seção quatro (4). A substituição de Seções Invariantes por traduções exige permissão especial de seus detentores dos direitos autorais, porém você pode incluir traduções de algumas ou todas as Seções Invariantes adicionalmente às versões originais dessas Seções Invariantes. Você pode incluir uma tradução desta Licença, e todos os avisos de licença no Documento, e quaisquer Declarações de Garantia, contanto que você inclua também a versão original em Inglês desta Licença e as versões originais daqueles avisos e declarações. No caso de uma divergência entre a tradução e a versão original desta Licença ou um aviso ou declaração, a versão original prevalecerá. Se uma seção no Documento for Intitulada “Agradecimentos”, “Dedicatórias”, ou “Histórico”, a exigência (seção 4) de Preservar seu Título (seção 1) tipicamente exigirá a modificação do título atual.

10. FINALIZAÇÃO

Você não pode copiar, modificar, sublicenciar, ou distribuir o Documento, exceto conforme expressamente provido sob esta Licença. Qualquer tentativa clandestina de copiar, modificar, sublicenciar, ou distribuir o Documento é inválida, e automaticamente finalizará seus direitos sob esta Licença. Entretanto, se você cessar todas as violações a esta Licença, então a sua licença oriunda de um detentor de direitos autorais em particular está restabelecida (a) provisoriamente, a menos e até que o detentor dos direitos autorais explicita e finalmente cancele sua licença; e (b) permanentemente, se o detentor dos direitos autorais falhar em notificar você da violação, por algum meio razoável, antes de sessenta (60) dias após a cessação. Além disso, a sua licença oriunda de um detentor de direitos autorais em particular está restabelecida permanentemente se o detentor dos direitos autorais notificar você sobre a violação por algum meio razoável, essa for a primeira vez que você recebeu um aviso de violação desta Licença (para qualquer trabalho) oriunda daquele detentor de direitos autorais, e você sanar a violação antes de decorridos trinta (30) dias após o seu recebimento do aviso. A finalização dos seus direitos sob esta seção não finaliza as licenças de varejistas que tenham recebido cópias ou direitos de você sob esta Licença. Se os seus direitos tiverem sido finalizados e não permanentemente restabelecidos, o recebimento de uma cópia de algum ou de tudo do mesmo material não concede a você direitos de utilizá-lo.

11. REVISÕES FUTURAS DESTA LICENÇA

A Free Software Foundation pode publicar novas, revisadas versões da Licença de Documentação Livre GNU de tempos em tempos. Tais novas versões serão similares na essência à presente versão, porém podem diferir em detalhes para abarcar novos problemas ou assuntos. Veja-se http://www.gnu.org/copyleft/. Para cada versão da Licença é dado um número distintivo de versão. Se o Documento especifica que uma versão numerada em particular desta Licença “ou qualquer versão posterior” se aplica a ele, você tem a opção de seguir os termos e condições ou da versão especificada ou de qualquer versão posterior que tenha sido publicada (não como um rascunho) pela Free Software Foundation. Se o Documento não especifica um número de versão desta Licença, você pode escolher qualquer versão já publicada (não como um rascunho) pela Free Software Foundation. Se o Documento especifica que um procurador pode decidir quais versões futuras desta Licença podem ser utilizadas, essa declaração pública do procurador de aceitação de uma versão permanentemente autoriza você a escolher aquela versão para o Documento.

12. RELICENCIAMENTO

“Sítio de Colaboração Massiva Multi autor” (ou “Sítio MMC”) significa qualquer servidor da Rede Mundial de Computadores que publica trabalhos sujeitos a direitos autorais e também provê facilidades proeminentes para qualquer pessoa editar esses trabalhos. Um wiki público que qualquer pessoa pode editar é um exemplo de tal servidor. Uma “Colaboração Massiva Multi autor” (ou “MMC”) contida no sítio significa qualquer conjunto de trabalhos sujeitos a direitos autorais assim publicados no sítio MMC. “CC-BY-SA” significa a licença Creative Commons Attribution-Share Alike 3.0 publicada pela Creative Commons Corporation, uma corporação sem fins lucrativos com seu domicílio empresarial situado em São Francisco, Califórnia, Estados Unidos da América do Norte, bem como versões futuras de esquerdos autorais dessa licença publicadas pela mesma organização. “Incorporar” significa publicar ou republicar um Documento, no todo ou em parte, como parte de outro Documento. Um MMC é “elegível para relicenciamento” se ele for licenciado sob esta Licença, e se todos os trabalhos que foram primeiro publicados sob esta Licença em algum lugar que não esse MMC, e subsequentemente incorporados, no todo ou em parte, no MMC, (1) não tinham textos de capa ou seções invariantes; e (2) estavam assim incorporados antes de 01 de novembro de 2008. O operador de um Sítio MMC pode republicar um MMC contido no sítio sob CC-BY-SA, no mesmo sítio, a qualquer tempo antes de 01 de agosto de 2009, contanto que o MMC seja elegível para relicenciamento.

ADENDO: Como utilizar esta Licença para seus documentos

Para utilizar esta Licença em um documento que você escreveu, inclua uma cópia da Licença no documento e coloque os seguintes avisos de direitos autorais e licença pouco depois da página de título:

```
Direitos autorais (C) ano  seu nome.
Permissão é concedida para copiar, distribuir e/ou modificar este
documento sob os termos da Licença de Documentação Livre GNU, Versão
1.3 ou qualquer versão posterior publicada pela Free Software
Foundation; sem Seções Invariantes, sem Textos de Primeira Capa, e sem
Textos de Quarta Capa.   Uma cópia da licença está inclusa na seção
intitulada ``Licença de Documentação Livre GNU''.
```

Se você tiver Seções Invariantes, Textos de Primeira Capa e Textos de Quarta Capa, substitua a linha ``sem ... Capa" por isto:

```
com as Seções Invariantes sendo liste seus títulos, com os
Textos de Primeira Capa sendo lista, e com os Textos de Quarta
Capa sendo lista.
```

Se você tiver Seções Invariantes sem Textos de Capa, ou alguma outra combinação dos três, mescle essas duas alternativas para adequar a situação.

Se o seu documento contém exemplos não triviais de código de programação, nós recomendamos publicar esses exemplos em paralelo, sob sua escolha de licença de software livre, tal como a Licença Pública Geral GNU, para permitir seu uso em software livre.

Índice Remissivo

Acl: 136
Attr: 135
Autoconf: 172
Automake: 174
Bash: 158
  tools: 61
Bash: 158
  tools: 61
Bc: 122
Binutils: 128
  tools, pass 1: 45
  tools, pass 2: 74
Binutils: 128
  tools, pass 1: 45
  tools, pass 2: 74
Binutils: 128
  tools, pass 1: 45
  tools, pass 2: 74
Bison: 156
  tools: 86
Bison: 156
  tools: 86
Bootscripts: 242
  usage: 253
Bootscripts: 242
  usage: 253
Bzip2: 113
Check: 193
Coreutils: 187
  tools: 62
Coreutils: 187
  tools: 62
DejaGNU: 127
Diffutils: 194
  tools: 63
Diffutils: 194
  tools: 63
E2fsprogs: 232
Eudev: 219
  configuring: 219
Eudev: 219
  configuring: 219
Expat: 163
Expect: 126
File: 118
  tools: 64
File: 118
  tools: 64
Findutils: 196
  tools: 65
Findutils: 196
  tools: 65
Flex: 123
Gawk: 195
  tools: 66
Gawk: 195
  tools: 66
GCC: 142
  tools, libstdc++ pass 1: 55
  tools, libstdc++ pass 2: 83
  tools, pass 1: 47
  tools, pass 2: 75
GCC: 142
  tools, libstdc++ pass 1: 55
  tools, libstdc++ pass 2: 83
  tools, pass 1: 47
  tools, pass 2: 75
GCC: 142
  tools, libstdc++ pass 1: 55
  tools, libstdc++ pass 2: 83
  tools, pass 1: 47
  tools, pass 2: 75
GCC: 142
  tools, libstdc++ pass 1: 55
  tools, libstdc++ pass 2: 83
  tools, pass 1: 47
  tools, pass 2: 75
GCC: 142
  tools, libstdc++ pass 1: 55
  tools, libstdc++ pass 2: 83
  tools, pass 1: 47
  tools, pass 2: 75
GDBM: 161
Gettext: 154
  tools: 85
Gettext: 154
  tools: 85
Glibc: 104
  tools: 52
Glibc: 104
  tools: 52