Linux From Scratch

Versão 7.5

Criado por Gerard Beekmans
Editado por Matthew Burgess e Bruce Dubbs

Linux From Scratch: Versão 7.5

por Criado por Gerard Beekmans e Editado por Matthew Burgess e Bruce Dubbs

Copyright © 1999-2014 Gerard Beekmans

Copyright © 1999-2014, Gerard Beekmans

Todos os diretos reservados.

Esta obra é licenciada sob os termos Creative Commons License.

Instruções de computador podem ser extraídas deste livros sob os termos da licença MIT License.

Linux® é uma marca registrada de Linus Torvalds.

Índice

Prefácio viii
i. Prefácio viii
ii. Público Alvo ix
iii. Arquiteturas ix
iv. LFS e Padrões x
v. Justificativa para os pacotes no Livro xi
vi. Pré-requisitos xvi
vii. Exigências de Sistema xvi
viii. Tipografia xix
ix. Estrutura xx
x. Errata xx
I. Introdução 1
1. Introdução 2
1.1. Como construir um sistema LFS 2
1.2. O que há de novo desde a última versão 3
1.3. Changelog (registro das mudanças) 4
1.4. Recursos 8
1.5. Ajuda 9
II. Preparando para Construção 12
2. Preparando uma nova partição 13
2.1. Introdução 13
2.2. Criando uma Nova Partição 13
2.3. Criando um Sistema de Arquivos na Partição 14
2.4. Montando a Nova Partição 15
3. Pacotes e Patches 17
3.1. Introdução 17
3.2. Todos os Pacotes 17
3.3. Patches Necessários 24
4. Preparações Finais 26
4.1. Sobre $LFS 26
4.2. Criando o diretório $LFS/tools 26
4.3. Adicionando o usuário LFS 27
4.4. Configurando o Ambiente 27
4.5. Sobre SBUs 29
4.6. Sobre as Ferramentas de Teste (Suites de Teste) 29
5. Construindo um Sistema Temporário 31
5.1. Introdução 31
5.2. Notas Técnicas sobre Toolchain (ferramentas para desenvolvimento do sistema) 31
5.3. Instruções gerais para compilação 33
5.4. Binutils-2.24 - Pass 1 35
5.5. GCC-4.8.2 - Pass 1 37
5.6. Linux-3.13.3 API Headers 40
5.7. Glibc-2.19 41
5.8. Libstdc++-4.8.2 44
5.9. Binutils-2.24 - Pass 2 46

5.10. GCC-4.8.2 - Pass 2 48
5.11. Tcl-8.6.1 52
5.12. Expect-5.45 54
5.13. DejaGNU-1.5.1 56
5.14. Check-0.9.12 57
5.15. Ncurses-5.9 58
5.16. Bash-4.2 59
5.17. Bzip2-1.0.6 60
5.18. Coreutils-8.22 61
5.19. Diffutils-3.3 62
5.20. File-5.17 63
5.21. Findutils-4.4.2 64
5.22. Gawk-4.1.0 65
5.23. Gettext-0.18.3.2 66
5.24. Grep-2.16 67
5.25. Gzip-1.6 68
5.26. M4-1.4.17 69
5.27. Make-4.0 70
5.28. Patch-2.7.1 71
5.29. Perl-5.18.2 72
5.30. Sed-4.2.2 73
5.31. Tar-1.27.1 74
5.32. Texinfo-5.2 75
5.33. Util-linux-2.24.1 76
5.34. Xz-5.0.5 77
5.35. Stripping (Esvaziando) 78
5.36. Mudando de Posse (Propriedade) 78
III. Construindo o Sistema LFS 79
6. Instalando os programas do Sistema básico 80
6.1. Introdução 80
6.2. Preparando sistemas de arquivo virtual do kernel 80
6.3. Gerenciamento de Pacote 82
6.4. Entrando no ambiente Chroot 85
6.5. Criando Diretórios 86
6.6. Criando Arquivos e Symlinks Essenciais 87
6.7. Linux-3.13.3 API Headers 90
6.8. Man-pages-3.59 91
6.9. Glibc-2.19 92
6.10. Ajustando a Toolchain 99
6.11. Zlib-1.2.8 101
6.12. File-5.17 102
6.13. Binutils-2.24 103
6.14. GMP-5.1.3 106
6.15. MPFR-3.1.2 108
6.16. MPC-1.0.2 109
6.17. GCC-4.8.2 110
6.18. Sed-4.2.2 115

6.19. Bzip2-1.0.6 116
6.20. Pkg-config-0.28 118
6.21. Ncurses-5.9 119
6.22. Shadow-4.1.5.1 122
6.23. Psmisc-22.20 125
6.24. Procps-ng-3.3.9 126
6.25. E2fsprogs-1.42.9 128
6.26. Coreutils-8.22 131
6.27. Iana-Etc-2.30 136
6.28. M4-1.4.17 137
6.29. Flex-2.5.38 138
6.30. Bison-3.0.2 140
6.31. Grep-2.16 141
6.32. Readline-6.2 142
6.33. Bash-4.2 144
6.34. Bc-1.06.95 146
6.35. Libtool-2.4.2 147
6.36. GDBM-1.11 148
6.37. Inetutils-1.9.2 149
6.38. Perl-5.18.2 151
6.39. Autoconf-2.69 154
6.40. Automake-1.14.1 156
6.41. Diffutils-3.3 158
6.42. Gawk-4.1.0 159
6.43. Findutils-4.4.2 160
6.44. Gettext-0.18.3.2 162
6.45. Groff-1.22.2 164
6.46. Xz-5.0.5 167
6.47. GRUB-2.00 169
6.48. Less-458 171
6.49. Gzip-1.6 172
6.50. IPRoute2-3.12.0 174
6.51. Kbd-2.0.1 176
6.52. Kmod-16 178
6.53. Libpipeline-1.2.6 180
6.54. Make-4.0 181
6.55. Patch-2.7.1 182
6.56. Sysklogd-1.5 183
6.57. Sysvinit-2.88dsf 184
6.58. Tar-1.27.1 185
6.59. Texinfo-5.2 186
6.60. Udev-208 (Extraído de systemd-208) 188
6.61. Util-linux-2.24.1 190
6.62. Man-DB-2.6.6 195
6.63. Vim-7.4 198
6.64. Sobre os Símbolos de depuração 201
6.65. Stripping (Esvaziando) Novamente 201

6.66. Limpando 202
7. Configurando os scripts de inicialização do sistema (Bootscripts) 203
7.1. Introdução 203
7.2. Configuração Geral de Rede 203
7.3. Customizando o Arquivo /etc/hosts 206
7.4. Manipulando dispositivos e módulos em um Sistema LFS 207
7.5. Criando Symlinks para Dispositivos 211
7.6. LFS-Bootscripts-20130821 214
7.7. Como esses scripts de inicialização funcionam? 216
7.8. Configurando o hostname do sistema 218
7.9. Configurando o Script Setclock 219
7.10. Configurando o Terminal Linux 219
7.11. Configurando o script sysklogd 222
7.12. O Arquivo rc.site 223
7.13. Os Arquivos de inicialização do Bash Shell 225
7.14. Criando o Arquivo /etc/inputrc 227
8. Tornando o Sistema LFS inicializável 229
8.1. Introdução 229
8.2. Criando o arquivo /etc/fstab 229
8.3. Linux-3.13.3 231
8.4. Usando o GRUB para Configurar o Processo de Boot 234
9. O final 236
9.1. O final 236
9.2. Seja Contado 236
9.3. Reiniciando o Sistema 236
9.4. O que fazer agora? 238
IV. Apêndices 239
A. Acrônimos e Abreviaturas 240
B. Agradecimentos 243
C. Dependências 246
D. Versão dos scripts de boot e sysconfig-20130821 259
D.1. /etc/rc.d/init.d/rc 259
D.2. /lib/lsb/init-functions 263
D.3. /etc/rc.d/init.d/functions 277
D.4. /etc/rc.d/init.d/mountvirtfs 291
D.5. /etc/rc.d/init.d/modules 292
D.6. /etc/rc.d/init.d/udev 294
D.7. /etc/rc.d/init.d/swap 295
D.8. /etc/rc.d/init.d/setclock 296
D.9. /etc/rc.d/init.d/checkfs 297
D.10. /etc/rc.d/init.d/mountfs 300
D.11. /etc/rc.d/init.d/udev_retry 301
D.12. /etc/rc.d/init.d/cleanfs 303
D.13. /etc/rc.d/init.d/console 305
D.14. /etc/rc.d/init.d/localnet 307
D.15. /etc/rc.d/init.d/sysctl 308
D.16. /etc/rc.d/init.d/sysklogd 309

D.17. /etc/rc.d/init.d/network 311
D.18. /etc/rc.d/init.d/sendsignals 312
D.19. /etc/rc.d/init.d/reboot 314
D.20. /etc/rc.d/init.d/halt 314
D.21. /etc/rc.d/init.d/template 315
D.22. /etc/sysconfig/modules 316
D.23. /etc/sysconfig/createfiles 317
D.24. /etc/sysconfig/udev-retry 317
D.25. /sbin/ifup 318
D.26. /sbin/ifdown 320
D.27. /lib/services/ipv4-static 322
D.28. /lib/services/ipv4-static-route 323
E. Regras de configuração do Udev 326
E.1. 55-lfs.rules 326
F. Licenças do LFS 327
F.1. Creative Commons License 327
F.2. The MIT License 331
Índice Remissivo 332

Prefácio

Prefácio

Minha jornada para aprender e entender melhor Linux começou há mais de uma década, 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 através dos anos. Algumas ainda existem, outras se transformaram em outra distribuição, e ainda há outras que ficaram relegadas à 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 diferentes 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 nós 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. 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 único 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* nasce.

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 à suas necessidades, o que é 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 divirta-se trabalhando no seu próprio Linux From Scratch e faça proveito dos benefícios de construir um sistema verdadeiramente seu.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Público Alvo

Há vários motivos que o levariam a ler este livro. Uma das questões que muitas pessoas levantam é, “por que passar por todo o sofrimento de construir manualmente um sistema Linux do Zero quando você pode simplesmente baixar e instalar um sistema existente?”

Uma razão importante que justifica a existência deste projeto é ajudá-lo a aprender como um sistema Linux funciona por dentro. Construir um sistema LFS ajuda a demonstrar o que faz um sistema Linux leve, e como tudo funciona e se inter-relaciona. Uma das melhores coisas que esta experiência de aprendizado pode proporcionar é a habilidade de customizar um sistema Linux para se adaptar as suas necessidades únicas.

Outro benefício chave do LFS é permitir que você tenha mais controle sobre o sistema sem ter que confiar na implementação de outra pessoa. Com LFS, você está no assento do motorista e dita cada aspecto do sistema.

LFS permite que você crie um sistema Linux muito compacto. Quando instalando uma distribuição regular, você é frequentemente forçado a instalar vários programas que provavelmente nunca serão usados ou entendidos. Esses programas desperdiçam recursos. Você pode argumentar que com os discos rígidos e as CPUs de hoje esses recursos não são mais um ponto a ser considerado. Algumas vezes, entretanto, você é limitado por questões de espaço se não por outros pontos. Pense sobre CDs inicializáveis, pendrives e sistemas embarcados. Essas são área onde LFS pode ser benéfico.

Outra vantagem de um sistema Linux customizado é a segurança. Compilando todo o sistema a partir do código fonte, você tem o poder de auditar tudo e aplicar todos as correções (patches) de segurança que quiser. Não é mais necessário esperar que alguém compile um programa que conserte uma falha de segurança. A menos que você examine o patch e implemente-o você mesmo, não há garantia de que o novo binário foi construído corretamente nem que resolve o problema.

O objetivo do LFS é construir um sistema base completo e utilizável. Se você não se interessa em construir seu sistema Linux do Zero, você não vai se beneficiar inteiramente das informações contidas neste livro.

Há muitas outras boas razões para construir seu próprio sistema LFS para listá-las aqui neste espaço limitado. No final, educação é de longe a razão mais forte. Conforme você prossegue com sua experiência, você vai descobrir a força que a informação e o conhecimento realmente trazem.

Arquiteturas

A principal arquitetura do LFS são os processadores AMD/Intel x86 (32 bits) e x86_64 (64 bits). Entretanto as instruções disponíveis neste livro funcionam, com algumas modificações, com o 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 um LFS previamente instalado, Ubuntu, Red Hat/Fedora, SuSE, ou outra distribuição que esteja disponível para a arquitetura que você tem. Note também que distribuições de 32-bits também podem ser instaladas e usadas em um computador com processador AMD/Intel 64-bits.

Alguns outros fatos sobre sistemas 64-bits precisam ser adicionados aqui. Quando comparado com sistemas 32-bits, o tamanho dos executáveis é um pouco maior e são um pouco mais rápidos. Por exemplo, em um teste de construção do LFS-6.5 em um processador Core2Duo os seguintes estatísticas foram verificadas:

```
Arquitetura  Tempo          Tamanho
32-bit       198.5 minutes  648 MB
64-bit       190.6 minutes  709 MB
```

Como você pode ver, a compilação no sistema 64-bit é apenas 4% mais rápida e 9% maior que a de 32-bit. Os ganhos de um sistema 64-bit são relativamente mínimos. Claro, se você tem mais de 4GB de RAM ou quer manipular dados que excedem 4GB, as vantagens de um sistema 64-bit são substanciais.

O padrão de 64-bit compilado que é resultante do LFS é considerando um sistema 64-bit "puro". Ou seja, ele apenas suporta executáveis 64-bit. Construir um sistema "mult-lib" requer a compilação de muitos aplicativos duas vezes, uma para 32-bit e outra para 64-bit. Não é dado suporte direto para isso porque esse procedimento iria interferir no objetivo educacional de prover instruções necessárias para um sistema Linux base limpo. Você pode procurar o projeto *Cross Linux From Scratch* para esse tópico avançado.

Este é um último comentário sobre sistemas 64-bit. Há pacotes antigos que não podem ser compilados em um sistema 64-bit "puro" ou necessitam de instruções específicas para isso. Geralmente esses pacotes tem instruções assembly 32-bit embutidas que falham quando compiladas em um sistema 64-bit. Isso inclui alguns drivers Xorg para algumas placas de vídeo antigas na página *http://xorg.freedesktop.org/releases/individual/driver/*. Muitos desses problemas podem ser resolvidos, mas podem necessitar de procedimentos específicos ou patches.

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 version 3.0 Draft 1 (FHS)*
- *Linux Standard Base (LSB) Specifications*

  O LSB tem cinco padrões separados: Core, C++, Desktop, Runtime Languages (linguagens em tempo de execução), e Printing(impressão). Em adição aos requerimentos genéricos há os requerimentos específicos de cada arquitetura. LFS tenta ficar de acordo com as arquiteturas discutidas na sessão anterior.

Nota

Muitas pessoas não concordam com as condições do LSB. O principal propósito de definir tais condições é garantir que softwares proprietários possam ser instalados e executados propriamente em um sistema que esteja de acordo com o referido padrão. Sendo o LFS baseado em código fonte, o usuário tem total controle sobre que pacotes ele quer e muitos escolhem não instalar alguns dos pacotes especificados pelo LSB.

Criar um sistema LFS capaz de passar nos testes para certificação LSB é possível, mas não sem a adição de pacotes que transcendem o escopo do LFS. Esses pacotes adicionais tem instruções para instalação no BLFS.

Pacotes disponibilizados pelo LFS que são necessários para satisfazer as condições 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 C++:	Gcc
LSB Desktop:	None
LSB Runtime Languages:	Perl
LSB Printing:	None
LSB Multimeda:	None

Pacotes disponibilizados pelo BLFS necessários para satisfazer os requerimentos do LSB

LSB Core:	At, Batch (uma parte de At), Cpio, Ed, Fcrontab, Initd-tools, Lsb_release, PAM, Sendmail (ou Postfix ou Exim)
LSB C++:	None
LSB Desktop:	ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Glib2, GTK+2, Icon-naming-utils, Libjpeg, Libpng, Libxml2, MesaLib, Pango, Qt4, Xorg
LSB Runtime Languages:	Python
LSB Printing:	CUPS
LSB Multimeda:	Alsa Libraries, NSPR, NSS, OpenSSL, Java, Xdg-utils

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

LSB Core:	time (executável) e pax
LSB C++:	None
LSB Desktop:	Qt3
LSB Runtime Languages:	None
LSB Printing:	None
LSB Multimeda:	None

Justificativa para os pacotes no Livro

Como dito anteriormente, o objetivo do LFS é construir um sistema em nível de base completo e funcional. Isso inclui todos os pacotes necessários para replicá-lo ao tempo que disponibiliza uma base relativamente pequena sobre a qual o usuário pode customizar um sistema mais completo baseado em suas próprias escolhas. Isso não significa que o LFS é o menor sistema possível. Vários pacotes importantes estão inclusos, mas não são estritamente necessários. A lista abaixo documenta a justificativa para cada pacote no livro.

- Autoconf

  Este pacote contém programas para produzir “shell scripts” que podem configurar automaticamente o código fonte a partir de um modelo. É geralmente necessário para reconstruir um pacote após atualizações nos procedimentos de construção do referido pacote.

- Automake

  Este pacote contém programas para gerar “Make files” a partir de um modelo. É geralmente necessário para reconstruir um pacote após atualizações nos procedimentos de construção do referido pacote.

- Bash

  Este pacote satisfaz um requerimento do LFS Core para disponibilizar uma interface Burn Shell para o sistema. Foi escolhido pelo seu uso comum e por suas capacidades que transcendem as funções básicas do shell.

- Bc

  Este pacote disponibiliza uma linguagem de processamento numérico com precisão arbitrária. Ele satisfaz requisitos necessários quando compilando o kernel Linux..

- Binutils

  Esse pacote contém um linker, um assembler e outras ferramentas para manipular “object files”. Os programas neste pacote são necessários para compilar a maioria dos pacotes no LFS além de muitos outros.

- Bison

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

- Bzip2

  Este pacote contém programas para compressão e descompressão de arquivos. É necessário para descomprimir vários pacotes no LFS.

- Check

  Este pacote contém um conjunto de ferramentas de teste para outros programas. É instalado apenas nas ferramentas temporárias.

- Coreutils

  Este pacote contém vários programas essenciais para visualização e manipulação de arquivos e diretórios. Esses pacotes são necessários para o gerenciamento de arquivos por linha de comando e para a instalação de todos os pacotes do LFS.

- DejaGNU

  Este pacote contém um sistema para testar outros programas. É instalado apenas nas ferramentas temporárias.

- Diffutils

  Este pacote contém programas que mostram as diferenças entre arquivos ou diretórios. Esses programas podem ser usados para criar patches e também são usados na construção de vários pacotes.

- E2fsprogs

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

- Expect

  Este pacote contém programas que executam scripts de diálogos com outros programas interativos. É comumente usado para testar outros pacotes. É instalado apenas nas ferramentas temporárias.

- File

  Este pacote é usado para determinar o tipo de um dado arquivo ou arquivos. Poucos pacotes precisam deles para serem construídos.

- Findutils

  Este pacote contém programas para encontrar arquivos em um sistema de arquivos. É usado em muitos scripts de construção.

- Flex

  Este pacote serve para gerar programas que reconhecem padrões em textos. É a versão GNU do programa lex (lexical analizer). É necessário para construir vários programas no LFS.

- Gawk

  Este pacote contém programas para manipular arquivos de texto. É a versão GNU do awk (Aho-Weinberg-Kernighan). É usado no script de construção de vários outros pacotes.

- Gcc

  Este pacote é o Gnu Compiler Collection. Ele contém os compiladores C e C++ assim como vários outros não construídos no LFS.

- GDBM

  Este pacote contém o GNU Database Manager Library. É usado por um outro pacote no LFS, o Man-DB.

- Gettext

  Este pacote contém utilitários e bibliotecas para internacionalização e localização de vários pacotes.

- Glibc

  Este pacote contém a biblioteca C principal. Programas Linux não funcionariam sem isso.

- GMP

  Este pacote contém bibliotecas matemáticas que contém funções úteis para aritmética de precisão arbitrária. É necessário para compilar Gcc.

- Grep

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

- Groff

  Este pacote contém programas para processamento e formatação de texto. Uma função importante desses programas é formatar páginas de manuais (man pages).

- GRUB

  Este é o Grand Unified Boot Loader. É um dos vários gerenciadores de inicialização disponíveis, mas é o mais flexível.

- Gzip

  Este pacote contém programas para compressão e descompressão de arquivos. É necessário para descomprimir vários pacotes no LFS.

- Iana-etc

  Este pacote provê dados para serviços e protocolos de rede. É necessário para habilitar suporte a rede adequado.

- Inetutils

  Este pacote contém programas para administração básica de rede.

- IProute2

  Este pacote contém programas básicos e avançados para redes IPv4 e IPv6. Foi escolhido em detrimento de outros pelo seu suporte a IPv6.

- Kbd

  Este pacote contém arquivos com tabelas de caracteres, utilitários para teclados que não são estadunidenses (non-US keyboards) e várias fontes.

- Kmod

  Este pacote contém programas para administrar os módulos do kernel do Linux.

- Less

  Este pacote contém um visualizador de textos muito bom que permite rolar o texto para cima ou para baixo. É também usado pelo Man-DB para visualizar páginas de manuais.

- Libpipeline

O pacote Libpipeline contém bibliotecas para manipular pipelines de sub-processos de uma maneira flexível e conveniente. É necessário para pacote Man-DB.

- Libtool

  Este pacote contém bibliotecas genéricas para suporte a scripts. Ele esconde a complexidade do uso de bibliotecas compartilhadas em uma interface consistente e portável. É necessário para as ferramentas de testes de outros pacotes do LFS.

- Linux Kernel

  Este pacote é o Sistema Operacional. É o Linux no ambiente GNU/Linux.

- M4

  Este pacote contém um processador de textos útil como ferramenta de construção para outros programas.

- Make

  Este pacote contém programas para direcionar a construção de pacotes. É necessário para quase todos os pacotes no LFS.

- Man-DB

  Este pacote contém programas para encontrar e visualizar páginas de manuais. Foi escolhido em detrimento ao pacote man devido a capacidades de internacionalização superiores. Ele faz as vezes do man.

- Man-pages

  Este pacote contém o conteúdo básico das páginas de manual do Linux.

- MPC

  Este pacote contém funções para aritmética de números complexos. É necessário para Gcc.

- MPFR

  Este pacote contém funções para aritmética de funções múltiplas. É necessário para Gcc.

- Ncurses

  Este pacote contém bibliotecas para manipulação de caracteres de forma independente do terminal. É utilizado geralmente para disponibilizar controle de cursor em um sistema com menus. É necessária para vários pacotes do LFS.

- Patch

  Este pacote contém um programa para modificar ou criar arquivos aplicando um arquivo *patch* tipicamente criado pelo programadiff. É necessário para construir vários pacotes LFS.

- Perl

  Este pacote é um interpretador para a linguagem PERL. É necessário para instalação e ferramentas de teste de vários pacotes no LFS.

- Pkg-config

  Este pacote contém um programa que retorna meta-dados sobre uma biblioteca ou pacote instalado.

- Procps-NG

  Este pacote contém programas para monitorar processos. Esses programas são úteis para administradores de sistemas, e são também usados pelos scripts de inicialização do LFS.

- Psmisc

  Este pacote contém programas para mostrar informações sobre processos que estão rodando. Esses programas são úteis para administradores de sistemas.

- Readline

  Este pacote é um conjunto de bibliotecas que oferecem capacidade para edição por linha de comando e histórico. É usado pelo Bash.

- Sed

  Este pacote permite a edição de texto sem abri-lo em um editor de textos. É também necessário para a maioria dos scripts de configuração dos pacotes do LFS.

- Shadow

  Este pacote contém programas para manipulação de senhas de uma maneira segura.

- Sysklogd

  Este pacote contém programas para registro de mensagens do sistema, tais como aqueles enviadas pelo kernel ou por daemons quando um evento não-usual acontece.

- Sysvinit

  Este pacote disponibiliza o programa init, o qual é “pai” de todos os outros processos no sistema Linux.

- Tar

  Este pacote provê capacidade de empacotamento e extração de virtualmente todos os pacotes usados no LFS.

- Tcl

  Este pacote contém a Linguagem de Comando de Ferramentas (Tool Command Language) usada por muitas ferramentas de teste nos pacotes do LFS. É instalado apenas nas ferramentas temporárias.

- Texinfo

  Este pacote contém programas para leitura, escrita e conversão de páginas info. É usado nos procedimentos de instalação de vários pacotes LFS.

- Udev

  Este pacote contém programas para a criação dinâmica de nós de dispositivos (device nodes). É uma alternativa à criação de milhares de dispositivos estáticos no diretório /dev.

- Util-linux

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

- Vim

  Este pacote contém um editor. Ele foi escolhido por sua compatibilidade com o clássico editor vi e o número gigante de habilidades poderosas. Um editor é uma escolha muito pessoal para muitos usuários e um outro editor poderia ser substituído se assim desejar.

- XZ Utils

  Este pacote contém programas para compressão e descompressão de arquivos. Ele tem a maior taxa de compressão geralmente disponível e é útil para descomprimir pacotes nos formatos XZ ou LZMA.

- Zlib

  Este pacote contém rotinas para compressão e descompressão usadas por alguns programas.

Pré-requisitos

Construir um sistema LFS não é uma tarefa simples. Esta tarefa requer um certo nível de conhecimento de administração de sistemas Unix para resolver problemas e executar as linhas de comandos listadas corretamente. Em particular, no mínimo, você 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 no 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 dos seguintes HOWTOs:

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

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

- The Linux Users' Guide *http://tldp.org/pub/Linux/docs/ldp-archived/users-guide/*

  Este guia cobre o uso de vários softwares do Linux. Esta referência também é antiga, mas ainda é válida.

- Dica essencial de pré-leitura *http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt*

  Essa é uma dica do LFS especialmente escrita para usuários novos no Linux. Ele inclui uma lista de links excelentes para fontes de informação sobre uma vasta gama de tópicos. Qualquer um tentando instalar LFS deveria ter um entendimento sobre muitos dos tópicos nessa dica.

Exigências de Sistema

O seu sistema deve ter os seguintes programas com as versões mínimas indicadas. Esse não deve ser um problema para a maioria das distribuições Linux modernas. Note também que muitas distribuições colocarão os cabeçalhos dos programas em pacotes separados, frequentemente na forma “<package-name>-devel” ou “<package-name>-dev”. Certifique-se de instalar tais pacotes se sua distribuição os fornece.

Versões anteriores dos programas listados podem funcionar, mas não foram testadas.

- **Bash-3.2** (/bin/sh deve ser um link simbólico ou um hard link para o bash)
- **Binutils-2.17** (Versões superiores a 2.24 não são recomendadas porque não foram testadas)
- **Bison-2.3** (/usr/bin/yacc sdeve ser um link para bison ou um pequeno script que o executa)
- **Bzip2-1.0.4**
- **Coreutils-6.9**
- **Diffutils-2.8.1**
- **Findutils-4.2.31**
- **Gawk-4.0.1** (/usr/bin/awk deve ser um link para gawk)
- **GCC-4.1.2** incluindo compilador C++, **g++** (Versões superiores a 4.8.2 não são recomendadas porque não foram testadas)

Nota

Em algumas distribuições, tem havido relatos que algumas bibliotecas usadas pelo gcc podem estar em um estado incosistente e que isso interfere com a construção do sistema LFS. Para verificar isso, procure em /usr/lib e possivelmente em /usr/lib64 por libgmp.la, libmpfr.la, e libmpc.la. Ou todas as três estão presentes ou ausentes, mas nunca apenas uma ou duas. Se esse problema existe em seu sistema, tanto pode-se renomear ou deletar os arquivos .la ou instalar os pacotes que faltam.

- **Glibc-2.5.1** (Versões superiores a 2.19 não são recomendadas porque não foram testadas)
- **Grep-2.5.1a**
- **Gzip-1.3.12**
- **Linux Kernel-2.6.32**

  A razão para o requisito da versão do kernel é que esta versão deve ser especificada quando compilando glibc no Capítulo 6 nas recomendações dos desenvolvedores. É também necessária para o udev.

  Se o kernel do sistema anfitrião é mais antigo que 2.6.32 você precisa atualiza o kernel por uma versão mais nova. Há dois caminhos a serem seguidos nessa situação. Primeiro, veja se seu fornecedor Linux disponibiliza pacotes de kernel 2.6.32 ou posteriores. Caso sim, então instale. Se o seu fornecedor não disponibiliza um pacote de kernel aceitável, ou você prefere não instalar, você pode compilar um kernel. Instruções para compilação do kernel e configuração do boot loader (considerando que o sistema anfitrião usa o GRUB) estão localizadas no Capítulo 8.
- **M4-1.4.10**
- **Make-3.81**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Sed-4.1.5**
- **Tar-1.18**
- **Xz-5.0.0**

Note que os links simbólicos (symlink) mencionados acima são necessários para construir um sistema LFS usando as instruções contidas neste livro. Symlinks que apontam para outros programas (como dash, mawk, etc.) podem funcionar, mas não foram testados ou não receberam suporte da equipe de desenvolvimento do LFS, e podem necessitar tanto de instruções adicionais ou patches para outros pacotes.

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

export LC_ALL=C
bash --version | head -n1 | cut -d" " -f2-4
echo "/bin/sh -> `readlink -f /bin/sh`"
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-
bison --version | head -n1
if [ -e /usr/bin/yacc ];
  then echo "/usr/bin/yacc -> `readlink -f /usr/bin/yacc`";
  else echo "yacc not found"; fi

bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
if [ -e /usr/bin/awk ];
  then echo "/usr/bin/awk -> `readlink -f /usr/bin/awk`";
  else echo "awk not found"; fi

gcc --version | head -n1
g++ --version | head -n1
ldd --version | head -n1 | cut -d" " -f2-  # glibc version
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
m4 --version | head -n1
make --version | head -n1
patch --version | head -n1
echo Perl `perl -V:version`
sed --version | head -n1
tar --version | head -n1
xz --version | head -n1

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

for lib in lib{gmp,mpfr,mpc}.la; do
  echo $lib: $(if find /usr/lib* -name $lib|
               grep -q $lib;then :;else echo not;fi) found
done
unset lib
EOF

bash version-check.sh
```

Tipografia

Para fazer as coisas mais fáceis de serem seguidas, há algumas convenções tipográficas usadas neste livro. Esta sessão contém alguns exemplos da formatação tipográfica encontrada neste livro.

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

Esta forma de texto deve ser digitada do jeito que está, a menos que seja dito o contrário no texto. 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'
```

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

Emphasis

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

http://www.linuxfromscratch.org/

Esta formatação é usada para hiperlinks tanto dentro da comunidade LFS quanto em páginas externas. Isso inclui HOWTOs, localizações de downloads e páginas da internet.

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

Esse formato é usado quando arquivos de configuração são criados. O primeiro comando diz para o sistema criar o arquivo `$LFS/etc/group` a partir do que quer 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>

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

[OPTIONAL TEXT]

Este formato é usado para encapsular texto que é opcional.

`passwd(5)`

Este formato é usado para referir-se a uma página de manual específica (man). O número entre parênteses indica uma seção específica dentro do manual. Por exemplo, **passwd** tem duas páginas de manual. Conforme as instruções de instalação do LFS, essas duas páginas 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** rá exibir a primeira página de manual que corresponde a “passwd”, a qual será /usr/share/man/man1/passwd.1. Para este exemplo em particular, você precisará executar o comando **man 5 passwd** para ler a página especifica que está sendo referenciada. Deve-se notar que a maioria das páginas de manual não possuem nomes de páginas duplicados em diferentes seções. Portanto, **man *<program name>*** geralmente é suficiente.

Estrutura

Este livro é dividido nas seguintes partes.

Parte I – Introdução

Parte I explica algumas notas importantes sobre como proceder com a instalação do LFS. Essa seção também fornece meta-informação sobre o livro.

Parte II – Preparando para Construção

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

Parte III – Construindo o Sistema LFS

Part III guia o leitor pela construção do sistema LFS—compilando e instalando todos os pacotes, um por um, configurando o script de inicialização e instalando o kernel. O sistema Linux resultante é a base sobre a qual outros programas podem ser construídos para expandir o sistema conforme desejado. No final deste livro, há uma lista de referência de fácil uso com todos os programas, bibliotecas e arquivos importantes que foram instalados.

Errata

Os programas utilizados para criar um sistema LFS são constantemente atualizados e melhorados. Avisos de segurança e correções de erros podem ser disponibilizados após a liberação do livro LFS. Para checar se versões de pacotes ou instruções nesta versão do LFS necessitam de modificações para acomodar vulnerabilidades ou corrigir erros, por favor visite *http://www.linuxfromscratch.org/lfs/errata/7.5/* antes de continuar com a construção do sistema. Você deve tomar nota de quaisquer mudanças e aplicá-las à seção relevante do livro conforme você constrói o 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, Mandriva, Red Hat ou SUSE). Esse sistema Linux existente (sistema anfitrião) será usado como ponto de partida provendo programas necessários, tais como um compilador linker e shell para a construção do novo sistema. Selecione a opção "desenvolvimento" durante a instalação do sistema anfitrião para poder acessar essas ferramentas.

Como uma alternativa para não instalar uma outra distribuição em sua máquina, você pode usar um LiveCD de uma distribuição comercial.

Capítulo 2 deste livro descreve como criar uma nova partição Linux nativa e um sistema de arquivos. É neste ponto que 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 estrutura 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 prosseguir seu trabalho através do Capítulo 5 e adiante.

Capítulo 5 explica o processo de instalação de vários pacotes que irão formar o conjunto de ferramentas para desenvolvimento (toolchain) que é usado para efetivamente construir o sistema no Capítulo 6. Alguns dos pacotes são necessários para resolver dependências recíprocas—por exemplo, para compilar um compilador, você precisa de um compilador.

Capítulo 5 também mostra como você deve construir uma primeira versão das ferramentas para desenvolvimento, o que inclui Binutils e GCC (primeira versão significa basicamente que esses dois pacotes centrais serão reinstalados). O próximo passo é construir Glibc, a biblioteca C. Glibc será compilada pelo conjunto de ferramentas construído na primeira versão. A partir daí uma segunda versão das ferramentas de desenvolvimento será construída. Desta vez, o conjunto de ferramentas será ligado dinamicamente à nova Glibc. O restante dos pacotes no Capítulo 5 são construídos usando a segunda versão das ferramentas de desenvolvimento. Uma vez feito isso, a instalação do LFS não será mais dependente do sistema anfitrião, com exceção do kernel que está funcionando.

Esse esforço para isolar o novo sistema do sistema anfitrião pode parecer excessivo. Uma explicação técnica completa sobre porque isso é feito é disponibilizada na Seção 5.2, "Notas Técnicas sobre Toolchain (ferramentas para desenvolvimento do sistema)".

No Capítulo 6, o sistema LFS completo é construído. O programa **chroot** (change root – mudar diretório raiz) é usado para entrar em um ambiente virtual e iniciar um shell cujo diretório raiz será a partição do LFS. Esse procedimento é muito similar à reinicializar e instruir o kernel a montar a partição do LFS como a partição raiz. O sistema não é reinicializado de fato, ele apenas **chroot (muda o diretório raiz)** porque criar um sistema inicializável requer trabalho adicional em cujo momento ainda não chegou. A maior vantagem é que usar o "chrooting" permite a você continuar usando o sistema anfitrião enquanto o LFS é construído. Enquanto espera por pacotes serem compilados, você pode continuar usando seu computador normalmente.

Para finalizar a instalação, os scripts de inicialização do LFS são configurados no Capítulo 7, e o kernel e gerenciador de boot são configurados no Capítulo 8. Capítulo 9 contém informações sobre como continuar a experiência com o LFS transcendendo este livro. Após os passos contidos no livro serem implementados, o computador estará pronto para reiniciar com o novo sistema LFS.

Este é o processo em poucas palavras. Informações detalhadas sobre cada passo são discutidas nos capítulos seguintes e na descrição dos pacotes. Itens que podem parecer complicados serão esclarecidos e tudo ficará em seu devido lugar conforme você embarca na aventura do LFS.

1.2. O que há de novo desde a última versão

Abaixo segue uma lista das atualizações de pacotes feitas deste a última versão do livro.

Atualizados para:

-
- Automake 1.14.1
- Binutils 2.24
- Bison 3.0.2
- Check 0.9.12
- Coreutils 8.22
- E2fsprogs 1.42.9
- File 5.17
- Flex 2.5.38
- GCC 4.8.2
- GDBM 1.11
- Gettext 0.18.3.2
- Glibc 2.19
- GMP 5.1.3
- Grep 2.16
- Inetutils 1.9.2
- IPRoute2 3.12.0
- Kbd 2.0.1
- Kmod 16
- Libpipeline 1.2.6
- Linux 3.13.3
- M4 1.4.17
- Make 4.0
- Man-DB 2.6.6
- Man-pages 3.59
- MPC 1.0.2
- Perl 5.18.2
- Tar 1.27.1
- TCL 8.6.1
- Texinfo 5.2

- Tzdata 2013i
- Udev 208 (Extraído de systemd-208)
- Util-Linux 2.24.1

Adicionados

•

- readline-6.2-fixes-2.patch

Removidos:

•

- automake-1.14-test-1.patch
- readline-6.2-fixes-1.patch
- texinfo-5.1-test-1.patch

1.3. Changelog (registro das mudanças)

Esta é a versão 7.5 do livro Linux From Scratch, de 2 de Março, 2014. Se este livro estiver com mais de seis meses, uma versão nova e melhor provavelmente já está disponível. Para descobrir, por favor verifique um dos sites: *http://www.linuxfromscratch.org/mirrors.html*.

Abaixo segue uma lista das mudanças de pacotes feitas deste a última versão do livro.

Changelog Entries:

- 2014-03-02
  - [bdubbs] - LFS-7.5 released.
  - [bdubbs] - Update host system requirements to address possible host installation of inconsistent libraries.
- 2014-02-18
  - [bdubbs] - Change kmod instructions to allow installation of man pages. Fixes *#3502*.
- 2014-02-16
  - [bdubbs] - Update to man-pages-3.5.9.
  - [bdubbs] - Incorporate beta FHS. Add /usr/share/ppd, /usr/libexec, /usr/share/color, /usr/local/share/color, /var/lib/color, and /usr/share/dict.
  - [bdubbs] - Incorporate beta FHS. Remove overrides for /usr/libexec: coreutils, findutils, gawk, gcc, glibc, inetutils, man-db, and tar. Also fixes *#3498*.
  - [bdubbs] - Incorporate beta FHS. Move grub sbin executables from /usr/sbin to /sbin.
  - [bdubbs] - Document two new glibc errors in the regression tests.
  - [bdubbs] - Move man-db after util-linux to satisfy a test dependency.
  - [bdubbs] - Update automake tests to accommodate util-linux in /tools and to speed the test up.
  - [bdubbs] - Restore building the flex static library.
- 2014-02-14
  - [bdubbs] - Make sed for omit-frame-pointers the same in Chapters 5 and 6. Fixes *#3497*.

- [bdubbs] - Simplify zimesone configuration in glibc. Thanks to Chris Staub for the patch. Fixes *#3496*.
- [bdubbs] - Let the glibc Makefile install rpc headers. Thanks to Chris Staub for the patch. Fixes *#3495*.
- [bdubbs] - Update to linux-3.13.3. Fixes *#3493*.

- 2014-02-13
  - [bdubbs] - Update to file-5.17. Fixes *#3491*.
  - [bdubbs] - Update to flex-2.5.38. Fixes *#3492*.
  - [bdubbs] - Update to man-pages-3.58. Fixes *#3490*.
- 2014-02-10
  - [bdubbs] - Update coreutils i18n patch. Thanks to Igor Izivkov for pointing it out. Fixes *#3488*.
- 2014-02-08
  - [bdubbs] - Update to glibc-2.19. Fixes *#3486*.
- 2014-02-07
  - [bdubbs] - Update to linux-3.13.2. Fixes *#3485*.
- 2014-02-05
  - [bdubbs] - Change expect library type in Chapter 5. Thanks to kammet for the report. Fixes *#3484*.
  - [bdubbs] - Fix e2fsprogs tests to run properly in the LFS chroot environment.
  - [bdubbs] - Remove unnecessary mkdir in groff.
- 2014-02-02
  - [bdubbs] - Update to linux-3.13.1. Fixes *#3483*.
- 2014-01-27
  - [bdubbs] - Add an environment variable to util-linux in Chapter 5 to prevent an installation error for hosts that have unneeded capabilities available.
- 2014-01-26
  - [bdubbs] - Update to man-pages-3.57. Fixes *#3480*.
  - [bdubbs] - Update to man-db-2.6.6. Fixes *#3479*.
  - [bdubbs] - Update to linux-3.13. Fixes *#3478*.
- 2014-01-25
  - [bdubbs] - Add a configure switch to util-linux in Chapter 5 to prevent an installation error for hosts that have systemd installed.
- 2014-01-22
  - [bdubbs] - Update to check-0.9.12. Fixes *#3477*.
  - [bdubbs] - Update to util-linux-2.24.1. Fixes *#3476*.
  - [bdubbs] - Update to mpc-1.0.2. Fixes *#3474*.
  - [bdubbs] - Update to man-pages-3.56. Fixes *#3470*.
  - [bdubbs] - Update to linux-3.12.7. Fixes *#3469*.
  - [bdubbs] - Update to perl-5.18.2. Fixes *#3465*.

- [bdubbs] - Update to gettext-0.18.3.2. Fixes *#3464*.

- 2014-01-21
  - [bdubbs] - Moved util-linux final build to be after udev. Fixed up e2fsprogs and udev to use the Chapter 5 build of util-linux. Fixes *#3467*.
- 2014-01-15
  - [bdubbs] - Added a Chapter 5 build of util-linux in preparation for moving the Chapter 6 build to after udev. This is not the complete fix as this build has not yet been incorporated into Chapter 6.
  - [bdubbs] - Mount /run as a tmpfs for Chapter 6.
- 2014-01-14
  - [bdubbs] - Update to inetutils-1.9.2 and remove reference to old BLFS page. Fixes *#3471* and *#3473*.
  - [bdubbs] - Fix hardcoded reference to /tools in Chapter 6 gcc. Fixes *#3466*.
  - [bdubbs] - Clean up /run and /tmp. Fixes *#3463*.
- 2014-01-02
  - [bdubbs] - Update to grep-2.16. Fixes *#3418*.
- 2013-12-29
  - [bdubbs] - Update to e2fsprogs-1.42.9. Fixes *#3462*.
  - [bdubbs] - Update to gdbm-1.11. Fixes *#3459*.
  - [bdubbs] - Update to kmod-16. Fixes *#3455*.
  - [bdubbs] - Update to automake-1.14.1. Fixes *#3458*.
  - [bdubbs] - Update readline patch to upstream level. Fixes *#3461*.
  - [bdubbs] - Use gcc version of libiberty.a. Fixes *#3456*.
  - [bdubbs] - Use different URL for shadow. Fixes *#3453*.
  - [bdubbs] - Update coreutils i18n patch to fix problem with uniq. Fixes *#3457*.
  - [bdubbs] - Remove no longer needed makeinfo from Host System Requirements. Fixes *#3460*.
- 2013-12-22
  - [matthew] - Update to Linux-3.12.6. Fixes *#3452*.
  - [matthew] - Update to Tzdata-2013i. Fixes *#3451*.
  - [matthew] - Update to Libpipeline-1.2.6. Fixes *#3449*.
  - [matthew] - Fix the coreutils-i18n patch, which introduced a regression in cut. Fixes *#3448*.
- 2013-12-16
  - [matthew] - Update to Coreutils-8.22. Fixes *#3447*.
  - [matthew] - Update to Man-Pages-3.55. Fixes *#3446*.
  - [matthew] - Update to Bison-3.0.2. Fixes *#3442*.
  - [matthew] - Update to Libpipeline-1.2.5. Fixes *#3440*.
  - [matthew] - Update to Binutils-2.24. Fixes *#3438*.
  - [matthew] - Update to File-5.16. Fixes *#3437*.

- [matthew] - Update to Linux-3.12.5. Fixes *#3436*.

- 2013-12-13
  - [bdubbs] - Fix kmod, procps-ng, zlib, readline, ncurses, and xz methods of establishing correct symbolic links for libraries.
  - [bdubbs] - Update to procps-ng-3.3.9. Fixes *#3439*.
  - [bdubbs] - Install non-essential programs from the xz package in /usr/bin. Fixes *#3445*.
- 2013-12-07
  - [bdubbs] - Enable building sulogin in util-linux. Suppress installing sysvinit's sulogin. Fixes *#3435*.
  - [bdubbs] - Suppress installing sysvinit's mesg and last that overwrite the versions installed by util-linux. Thanks to Chris Staub. Fixes *#3434*.
  - [bdubbs] - Add a sed to diffutils so locales are properly installed. Fixes *#3433*.
  - [bdubbs] - Updates to the installed programs lists for several packages. Thanks to Chris Staub. Fixes *#3432*.
  - [bdubbs] - Fix location of binaries and libraries for kmod and xz. Fixes *#3443*.
- 2013-11-23
  - [matthew] - Update to IPRoute2-3.12.0. Fixes *#3431*.
  - [matthew] - Update to Linux-3.12.1. Fixes *#3428*.
  - [matthew] - Update to Tar-1.27.1. Fixes *#3425*.
  - [matthew] - Update to Bison-3.0.1. Fixes *#3423*.
  - [matthew] - Update to Check-0.9.11. Fixes *#3422*.
- 2013-11-05
  - [matthew] - Update to Linux-3.12. Fixes *#3421*.
  - [matthew] - Update to Kbd-2.0.1. Fixes *#3420*.
  - [matthew] - Update to Tzdata-2013h. Fixes *#3416*.
- 2013-11-04
  - [bdubbs] - Disable pkg-config lookups in the Chapter 5 check program that may cause unwanted host system libraries to be linked into check.
- 2013-10-21
  - [bdubbs] - Update to util-linux-2.24. Fixes *#3415*.
- 2013-10-19
  - [matthew] - Update to Linux-3.11.6. Fixes *#3414*.
- 2013-10-18
  - [matthew] - Update to GCC-4.8.2. Fixes *#3413*.
- 2013-10-15
  - [matthew] - Update to Linux-3.11.5. Fixes *#3411*.
- 2013-10-14
  - [matthew] - Update to Make 4.0. Fixes *#3410*.

    - [matthew] - Update to Tar 1.27. Fixes *#3409*.
- 2013-10-08
    - [matthew] - Update stylesheets to docbook-xsl-1.78.1.
- 2013-10-06
    - [matthew] - Use xz version of M4 tarball.
    - [matthew] - Update to Linux 3.11.4. Fixes *#3408*.
- 2013-10-02
    - [bdubbs] - Update to Udev 208 (extracted from systemd-208). Fixes *#3406*.
    - [bdubbs] - Update to tzdata-2013g. Fixes *#3400*.
    - [bdubbs] - Update to File-5.15. Fixes *#3402*.
    - [bdubbs] - Update to linux-3.11.3. Fixes *#3403*.
    - [bdubbs] - Update to texinfo-5.2. Fixes *#3404*.
    - [bdubbs] - Update to gmp-5.1.3. Fixes *#3405*.
- 2013-09-23
    - [bdubbs] - Update to man-pages-3.54. Fixes *#3398*.
    - [bdubbs] - Update to tcl-8.6.1. Fixes *#3399*.
    - [bdubbs] - Update to M4-1.4.17. Fixes *#3401*.
- 2013-09-15
    - [matthew] - Update to Linux-3.11.1. Fixes *#3397*.
    - [matthew] - Update to Kbd-2.0.0. Fixes *#3390*.
- 2013-09-13
    - [bdubbs] - Update to systemd-207. Fixes *#3396*.
- 2013-09-10
    - [bdubbs] - Update to gettext-0.18.3.1. Fixes *#3389*.
    - [bdubbs] - Update to kmod-15. Fixes *#3392*.
    - [bdubbs] - Update to iproute2-3.11.0. Fixes *#3395*.
    - [bdubbs] - Update to linux-3.11.0. Fixes *#3394*.
- 2013-09-08
    - [bdubbs] - LFS-7.4 released.

1.4. Recursos

1.4.1. FAQ (Perguntas Frequentes)

Se durante a construção do LFS você encontrar quaisquer erros, tiver perguntas, ou achar que há um erro de impressão no livro, por favor comece verificando no FAQ (Frequentely Asked Questions – perguntas feitas com frequência) que está localizado em *http://www.linuxfromscratch.org/faq/*.

1.4.2. Lista de Email

O servidor `linuxfromscratch.org` hospeda várias listas de discussão usadas para o desenvolvimento do projeto LFS. Essas listas incluem a lista principal de desenvolvimento e a lista de suporte, dentre outras. Se o FAQ não resolver seu problema, o próximo passo seria procurar na lista de discussão no endereço *http://www.linuxfromscratch.org/search.html*.

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

1.4.3. IRC

Vários membros da comunidade LFS oferecem assistência no IRC (Internet Relay Chat). Antes de usar este suporte, por favor certifique-se de que sua pergunta não está no FAQ do LFS ou nos arquivos das listas de discussão. Você pode encontrar a rede IRC no endereço `irc.freenode.net`. O nome do canal de suporte é #LFS-support.

1.4.4. Espelhos da Página (Mirror Sites)

O projeto LFS tem uma série de espelhos pelo mundo para fazer com que o acesso ao site do projeto e o download dos pacotes seja mais conveniente. Por favor visite o site do LFS *http://www.linuxfromscratch.org/mirrors.html* para uma lista dos espelhos que estão no ar.

1.4.5. Informações para Contatos

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

1.5. Ajuda

Se um problema ou questão for encontrado durante o trabalho com este livro, por favor verifique o FAQ na página *http://www.linuxfromscratch.org/faq/#generalfaq*. Com frequência, questões já estão respondidas nesta página. Se sua questão não está respondida nesta página, por favor tente econtrar a fonte do problema. A dica seguinte vai lhe fornecer orientações com relação a resolução de problemas: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

Se você não pode achar seu problema no FAQ, procure nas listas de discussão através da página *http://www.linuxfromscratch.org/search.html*.

Nós também temos uma comunidade maravilhosa que está disposta a oferecer assistência através das listas de discussão e do IRC (veja Seção 1.4, “Recursos” deste livro). Entretanto, nós temos várias questões todos os dias e várias delas podem ser facilmente respondidas indo primeiro ao FAQ ou procurando nas listas de discussão. Então, para que nós possamos oferecer a melhor assistência possível, você precisa pesquisar um pouco por contra própria. Isso nos permite manter o foco nas necessidades menos usuais de suporte. Se sua busca não produz solução, por favor inclua toas as informações relevantes (mencionadas abaixo) no seu pedido de ajuda.

1.5.1. Itens a serem Mencionados

Além de uma breve explanação do problema, os itens essenciais a serem incluídos na requisição de ajuda são:

- A versão do livro sendo usado (neste caso 7.5)
- O sistema anfitrião e sua versão sendo usado para criar o LFS
- A saída da Seção vii.1, “ ”

- O pacote ou seção onde o problema foi encontrado
- A mensagem de erro exata ou o sintoma recebido
- Cite se você desviou ou não das instruções do livro

Nota

Desviar-se das instruções deste livro *não* significa que nós não vamos ajudá-lo. Afinal de contas, LFS está relacionado a preferências pessoais. Ser sincero sobre quaisquer mudanças nos procedimentos nos ajudará a avaliar e determinar as possíveis causas do seu problema.

1.5.2. Problemas com os scripts de Configuração

Se algo der errado quando executar o script **configure** revise o arquivo `config.log`. This file may contain errors encountered during **configure** which were not printed to the screen. Inclua as linhas *relevantes* caso você precise de ajuda.

1.5.3. Problemas com Compilação

Tanto a saída da tela quando o conteúdo de vários arquivos são úteis para determinar a causa de problemas durante a compilação. A saída da tela do script **configure** e do **make** podem ser úteis. Não é necessário incluir toda a saída, mas inclua informações relevantes suficiente. Abaixo segue exemplo do tipo de informação a incluir da saída do comando **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
```

Neste caso, muitas pessoas incluiriam apenas a seção final:

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

Isso não é o suficiente para diagnosticar propriamente o problema, porque essa linha apenas mostra que algo deu errado, não *o quê* deu errado. Toda a seção, como no exemplo acima, é o que deveria ser salvo porque ela inclui o comando que foi executado e a(s) mensagem de erro associada.

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

Parte II. Preparando para Construção

Capítulo 2. Preparando uma nova partição

2.1. Introdução

Neste capítulo, a partição que recebe o sistema LFS é preparada. Nós criaremos a partição, o sistema de arquivos e montaremos o mesmo.

2.2. Criando uma Nova Partição

Como muitos outros sistemas operacionais, LFS é geralmente instalado em uma partição dedicada. A abordagem recomendada para construir um sistema LFS é usar uma partição vazia ou, se você tem espaço suficiente em disco, criar uma nova partição.

Um sistema mínimo necessita de uma partição com cerca de 2.8 gigabytes (GB). Isso é espaço suficiente para armazenar e compilar todos os pacotes. Entretanto, se você pretende usar o sistema LFS como sistema Linux principal, mais programas provavelmente serão instalados e você precisará de mais espaço. Uma partição de 10GB é um tamanho razoável que permite crescimento do sistema. O sistema LFS em si não vai usar muito espaço. Uma boa parte desse espaço requerido é para proporcionar espaço para armazenamento temporário. A compilação de pacotes pode necessitar de muito espaço de disco que será recuperado após a instalação dos pacotes.

Devido ao fato de nem sempre haver memória RAM (Random Access Memory – Memória de acesso aleatório) suficiente para os processos de compilação, seria uma boa idéia usar uma pequena partição como `swap`. É usada pelo kernel para armazenar dados raramente utilizados deixando mais memória disponível para os processos ativos. A partição `swap` para o sistema LFS pode ser a mesma usada pelo sistema anfitrião, nesse caso não seria necessário criar outra partição.

Inicie um programa de particionamento de disco como **cfdisk** ou **fdisk** com a opção em linha de comando indicando o disco rígido no qual a partição será criada—por exemplo `/dev/sda` para o primeiro disco IDE (Integrated Drive Electronics). Create a Linux native partition and a `swap` partition, if needed. Por favor, recorra ao `cfdisk(8)` ou `fdisk(8)` se você ainda não sabe como usar esses programas.

Nota

Para usuários experientes, é possível usar outros esquemas de partição. O novo sistema LFS pode ser instalado em um *RAID* ou em um volume lógico *LVM*. Entretanto, algumas dessas opções requerem um *initramfs*, o que é um tópico avançado. Esses métodos de partição não são recomendados para quem usa o LFS pela primeira vez.

Lembre-se da designação da nova partição (e.g., `sda5`). Este livro irá se referir a esta partição como partição LFS. Lembre-se também da designação da partição `swap`. Esses nomes serão necessários posteriormente para o arquivo `/etc/fstab`.

2.2.1. Outros assuntos relacionados a Partições

Pedidos de ajuda com relação a particionamento de disco são frequentemente enviados à lista de correio do LFS. Esse é um assunto muito subjetivo. O padrão para a maioria das distribuições é usar todo o disco, exceto uma pequena partição para swap. Isso não é ideal para o LFS por vários motivos. Isso reduz flexibilidade, faz com que o compartilhamento de dados entre múltiplas distribuições ou entre sistemas LFS seja mais difícil, faz com que backups consumam mais tempo, e podem desperdiçar espaço de disco devido à alocação indeficiente do sistema de arquivos.

2.2.1.1. A partição raiz (root partition)

Uma partição raiz no sistema LFS (não confundir partição raiz (root partition) / com diretório `/root`) de dez gigabytes é uma boa escolha para muitos sistemas. Ela disponibiliza espaço suficiente para construir o LFS e muito do BLFS, mas é pequena o suficiente para permitir que outras partições sejam criadas facilmente para experimentação.

2.2.1.2. A Partição Swap

A maioria das distribuições automaticamente cria uma partição swap. Geralmente o tamanho recomendado da swap é o dobro da memória RAM, entretanto isso raramente é necessário. Se há limitações com relação a espaço de disco, mantenha a partição swap com 2GB e monitore o a quantidade de memória swap que é consumida.

O uso da memória swap nunca é uma coisa boa. Geralmente você pode dizer se o sistema está usando a swap simplesmente prestando atenção na atividade de disco e observando como o sistema reage a comandos. A primeira reação em caso de uso da swap deve ser verificar o uso irracional de algum comando, por exemplo a tentativa de editar um arquivo de 5GB. Se o uso da memória swap se tornar uma ocorrência normal, a melhor solução é comprar mais memória RAM para seu sistema.

2.2.1.3. Partições de Conveniência

Há várias outras partições que não são necessárias, mas deveriam ser consideradas quando do design do layout do disco. A lista seguinte não é abrangente, mas serve como um guia.

- /boot – Altamente recomendada. Use essa partição para armazenar o kernel e outras informações relacionadas ao boot. Para minimizar potenciais problemas de inicialização com discos grandes, maça dessa partição a primeira na partição física no seu primeiro disco. O tamanho de 100MB é adequado.
- /home – Altamente recomendada. Compartilhe seu diretório de usuário (home) e personalizações entre múltiplas distribuições ou sistemas LFS. O tamanho é geralmente grande e depende do espaço de disco disponível.
- /usr – Uma partição /usr separada é geralmente usada quando se disponibiliza um servidor para um cliente com pouca demanda ou uma estação de trabalho sem disco. Normalmente não é necessário para o LFS. O tamanho de 5GB vai servir para a maioria das instalações.
- /opt – Esse diretório é mais útil para o BLFS onde múltiplas instalações de pacotes grandes como Gnome ou KDE podem ser instalados sem embutir os arquivos na hierarquia da pasta /usr. Se usado, 5 a 10GB geralmente é suficiente.
- /tmp – Um diretório /tmp separado é raro, mas útil se configurando um cliente com poucos recursos. Esta partição, se usada, geralmente não precisará exceder 2GB.
- /usr/src – Esta partição é muito útil para disponibilizar uma localização para armazenar os arquivos-fontes do BLFS e compartilhá-los entre os sistemas LFS. Também pode ser utilizado como um local para construção dos pacotes BLFS. Uma partição razoavelmente grande de 30-50GB é o suficiente.

Quaisquer partições separadas que você queira que sejam montadas durante o boot precisam ser especificadas no arquivo `/etc/fstab`. Detalhes sobre como especificar partições serão discutidos na Seção 8.2, “Criando o arquivo /etc/fstab”.

2.3. Criando um Sistema de Arquivos na Partição

Agora que uma partição em branco foi construída, 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 do tamanho da partição. Por exemplo:

ext2
: é adequado para partições pequenas que são atualizadas com pouca frequencia tais como /boot.

ext3
: é uma atualização do ext2 que inclui journal para ajudar a recuperar status de partições 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. Ela dispõe város novos recursos incluindo timestamp em nano-segundos, criação e uso de arquivos muito grandes (16 TB), e melhoramento de velocidade.

Outros sistemas de arquivos, incluindo FAT32, NTFS, ReiserFS, JFS, e XFS são úteis para propósitos específicos. Mais informações sobre esses sistemas de arquivos podem ser encontradas em *http://en.wikipedia.org/wiki/Comparison_of_file_systems*.

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

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

Se você está usando uma partição `swap` existente, não há necessidade de formatá-la. Se uma nova partição `swap` foi criada, ela deve ser inicializada com este comando:

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

Substitua <*yyy*> com o nome da partição `swap`.

2.4. 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 arbitrário. Para os propósitos deste livro, assume-se que o sistema de arquivos está montado na pasta `/mnt/lfs`, mas cabe a você escoher o diretório.

Escolha um ponto de montagem e atribua o mesmo à variável de ambiente `LFS` executando:

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

A seguir, crie o ponto de montagem e monte o sistema de arquivos do 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 o LFS (e.g., uma para `/` e outra para `/usr`), monte as partições usando:

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

Substitua <*xxx*> e <*yyy*> com os nomes das partições apropriados.

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

Se você estiver usando uma partição swapassegure-se de que ela está habilitada com o comando **swapon**:

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

Substitua *<zzz>* com o nome da partição swap.

Agora que há um lugar estabelecido para o trabalho, é hora 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 software cujo funcionamento foi testado e nas quais este livro é baseado. Nós recomendamos veementemente que não se utilize versões novas, uma vez que os comandos para construção de uma versão podem não funcionar em uma nova versão. A nova versão do pacote também pode ter problemas que precisem ser corrigidos. Essas correções serão desenvolvidas e estabilizadas na versão do livro que está em fase de desenvolvimento.

Links de downloads nem sempre podem estar acessíveis. Se a localização de uma página de download mudou desde a publicação deste livro, o Google (*http://www.google.com/*) disponibiliza uma ferramenta de busca útil para localizar a maioria dos pacotes. Se a busca não for bem sucedida, tente um dos meios alternativos de download discutidos na página *http://www.linuxfromscratch.org/lfs/packages.html#packages*.

Pacotes e pactches baixados precisam ser armazenados em algum lugar que esteja convenientemente disponível durante todo o procedimento de construção do sistema. Um diretório de trabalho também se faz necessário para extrair as fontes e contruí-las. `$LFS/sources` pode ser usado como um local para armazenar os arquivos compactados e patches e como diretório de trabalho. Usando este diretório, os elementos necessários estarão localizados na partição LFS e estarão disponíveis durante todos os estágios do processo de construção.

Para criar esse diretório, execute o comando seguinte, como usuário `root`, antes de começar a seção download:

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

Faça esse diretório livre para escrita e "sticky". “Sticky” siginifica que mesmo que este diretório esteja disponível para que múltiplos usuários tenham permissão de escrita, só o dono de um arquivo pode deletar esse arquivo. O comando seguinte habilitará o modo de escrita e o sticky:

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

Uma maneira fácil de baixar todos os pacotes e patches é utilizar uma lista *wget-list* como entrada para o comando **wget**. Por exemplo:

```
wget -i wget-list -P $LFS/sources
```

Adicionalmente, começando com o LFS-7.0, há um arquivo separado, *md5sums*, que pode ser usado para verificar se todos os pacotes corretos estão disponíveis antes de prosseguir. Coloque o arquivo no diretório `$LFS/sources` e execute os comandos:

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

3.2. Todos os Pacotes

Baixe ou obtenha de outra forma os seguintes pacotes:

• **Autoconf (2.69) - 1,186 KB:**
Home page: *http://www.gnu.org/software/autoconf/*
Download: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.69.tar.xz*
MD5 sum: `50f97f4159805e374639a73e2636f22e`

• Automake (1.14.1) - 1,456 KB:
Home page: *http://www.gnu.org/software/automake/*
Download: *http://ftp.gnu.org/gnu/automake/automake-1.14.1.tar.xz*
MD5 sum: `7fc29854c520f56b07aa232a0f880292`

• Bash (4.2) - 6,845 KB:
Home page: *http://www.gnu.org/software/bash/*
Download: *http://ftp.gnu.org/gnu/bash/bash-4.2.tar.gz*
MD5 sum: `3fb927c7c33022f1c327f14a81c0d4b0`

• Bc (1.06.95) - 288 KB:
Home page: *http://www.gnu.org/software/bc/*
Download: *http://alpha.gnu.org/gnu/bc/bc-1.06.95.tar.bz2*
MD5 sum: `5126a721b73f97d715bb72c13c889035`

• Binutils (2.24) - 22,184 KB:
Home page: *http://www.gnu.org/software/binutils/*
Download: *http://ftp.gnu.org/gnu/binutils/binutils-2.24.tar.bz2*
MD5 sum: `e0f71a7b2ddab0f8612336ac81d9636b`

• Bison (3.0.2) - 1,882 KB:
Home page: *http://www.gnu.org/software/bison/*
Download: *http://ftp.gnu.org/gnu/bison/bison-3.0.2.tar.xz*
MD5 sum: `146be9ff9fbd27497f0bf2286a5a2082`

• Bzip2 (1.0.6) - 764 KB:
Home page: *http://www.bzip.org/*
Download: *http://www.bzip.org/1.0.6/bzip2-1.0.6.tar.gz*
MD5 sum: `00b516f4704d4a7cb50a1d97e6e8e15b`

• Check (0.9.12) - 714 KB:
Home page: *http://check.sourceforge.net/*
Download: *http://sourceforge.net/projects/check/files/check/0.9.12/check-0.9.12.tar.gz*
MD5 sum: `46fe540d1a03714c7a1967dbc6d484e7`

• Coreutils (8.22) - 5,210 KB:
Home page: *http://www.gnu.org/software/coreutils/*
Download: *http://ftp.gnu.org/gnu/coreutils/coreutils-8.22.tar.xz*
MD5 sum: `8fb0ae2267aa6e728958adc38f8163a2`

• DejaGNU (1.5.1) - 566 KB:
Home page: *http://www.gnu.org/software/dejagnu/*
Download: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.5.1.tar.gz*
MD5 sum: `8386e04e362345f50ad169f052f4c4ab`

• Diffutils (3.3) - 1,170 KB:
Home page: *http://www.gnu.org/software/diffutils/*
Download: *http://ftp.gnu.org/gnu/diffutils/diffutils-3.3.tar.xz*
MD5 sum: `99180208ec2a82ce71f55b0d7389f1b3`

• E2fsprogs (1.42.9) - 5,928 KB:
Home page: *http://e2fsprogs.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.42.9.tar.gz*
MD5 sum: `3f8e41e63b432ba114b33f58674563f7`

• Expect (5.45) - 614 KB:
Home page: *http://expect.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/expect/expect5.45.tar.gz*
MD5 sum: `44e1a4f4c877e9ddc5a542dfa7ecc92b`

• File (5.17) - 694 KB:
Home page: *http://www.darwinsys.com/file/*
Download: *ftp://ftp.astron.com/pub/file/file-5.17.tar.gz*
MD5 sum: `e19c47e069ced7b01ccb4db402cc01d3`

Nota

File (5.17) pode não estar disponível na localização listada. Os administradores do site ocasionalmente removem versões antigas quando novas são liberadas. Uma localização de download alternativa que pode ter a versão correta pode ser encontrada na pagina: *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.4.2) - 2,100 KB:
Home page: *http://www.gnu.org/software/findutils/*
Download: *http://ftp.gnu.org/gnu/findutils/findutils-4.4.2.tar.gz*
MD5 sum: `351cc4adb07d54877fa15f75fb77d39f`

• Flex (2.5.38) - 1,590 KB:
Home page: *http://flex.sourceforge.net*
Download: *http://prdownloads.sourceforge.net/flex/flex-2.5.38.tar.bz2*
MD5 sum: `b230c88e65996ff74994d08a2a2e0f27`

• Gawk (4.1.0) - 2,004 KB:
Home page: *http://www.gnu.org/software/gawk/*
Download: *http://ftp.gnu.org/gnu/gawk/gawk-4.1.0.tar.xz*
MD5 sum: `b18992ff8faf3217dab55d2d0aa7d707`

• GCC (4.8.2) - 83,984 KB:
Home page: *http://gcc.gnu.org/*
Download: *http://ftp.gnu.org/gnu/gcc/gcc-4.8.2/gcc-4.8.2.tar.bz2*
MD5 sum: `a3d7d63b9cb6b6ea049469a0c4a43c9d`

• GDBM (1.11) - 796 KB:
Home page: *http://www.gnu.org/software/gdbm/*
Download: *http://ftp.gnu.org/gnu/gdbm/gdbm-1.11.tar.gz*
MD5 sum: `72c832680cf0999caedbe5b265c8c1bd`

• Gettext (0.18.3.2) - 15,810 KB:
Home page: *http://www.gnu.org/software/gettext/*
Download: *http://ftp.gnu.org/gnu/gettext/gettext-0.18.3.2.tar.gz*
MD5 sum: `241aba309d07aa428252c74b40a818ef`

• Glibc (2.19) - 11,801 KB:
Home page: *http://www.gnu.org/software/libc/*
Download: *http://ftp.gnu.org/gnu/glibc/glibc-2.19.tar.xz*
MD5 sum: `e26b8cc666b162f999404b03970f14e4`

• GMP (5.1.3) - 1,777 KB:
Home page: *http://www.gnu.org/software/gmp/*
Download: *http://ftp.gnu.org/gnu/gmp/gmp-5.1.3.tar.xz*
MD5 sum: `e5fe367801ff067b923d1e6a126448aa`

• Grep (2.16) - 1,184 KB:
Home page: *http://www.gnu.org/software/grep/*
Download: *http://ftp.gnu.org/gnu/grep/grep-2.16.tar.xz*
MD5 sum: `502350a6c8f7c2b12ee58829e760b44d`

• Groff (1.22.2) - 3,926 KB:
Home page: *http://www.gnu.org/software/groff/*
Download: *http://ftp.gnu.org/gnu/groff/groff-1.22.2.tar.gz*
MD5 sum: `9f4cd592a5efc7e36481d8d8d8af6d16`

• GRUB (2.00) - 5,016 KB:
Home page: *http://www.gnu.org/software/grub/*
Download: *http://ftp.gnu.org/gnu/grub/grub-2.00.tar.xz*
MD5 sum: `a1043102fbc7bcedbf53e7ee3d17ab91`

• Gzip (1.6) - 712 KB:
Home page: *http://www.gnu.org/software/gzip/*
Download: *http://ftp.gnu.org/gnu/gzip/gzip-1.6.tar.xz*
MD5 sum: `da981f86677d58a106496e68de6f8995`

• Iana-Etc (2.30) - 201 KB:
Home page: *http://freshmeat.net/projects/iana-etc/*
Download: *http://anduin.linuxfromscratch.org/sources/LFS/lfs-packages/conglomeration//iana-etc/iana-etc-2.30.tar.bz2*
MD5 sum: `3ba3afb1d1b261383d247f46cb135ee8`

• Inetutils (1.9.2) - 2,188 KB:
Home page: *http://www.gnu.org/software/inetutils/*
Download: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.9.2.tar.gz*
MD5 sum: `aa1a9a132259db83e66c1f3265065ba2`

• IPRoute2 (3.12.0) - 415 KB:
Home page: *http://www.kernel.org/pub/linux/utils/net/iproute2/*
Download: *http://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-3.12.0.tar.xz*
MD5 sum: `f87386aaaecafab95607fd10e8152c68`

• Kbd (2.0.1) - 1,962 KB:
Home page: *http://ftp.altlinux.org/pub/people/legion/kbd*
Download: *http://ftp.altlinux.org/pub/people/legion/kbd/kbd-2.0.1.tar.gz*
MD5 sum: `cc0ee9f2537d8636cae85a8c6541ed2e`

• Kmod (16) - 1,408 KB:
Download: *http://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-16.tar.xz*
MD5 sum: `3006a0287211212501cdfe1211b29f09`

• Less (458) - 308 KB:
Home page: *http://www.greenwoodsoftware.com/less/*
Download: *http://www.greenwoodsoftware.com/less/less-458.tar.gz*
MD5 sum: `935b38aa2e73c888c210dedf8fd94f49`

• LFS-Bootscripts (20130821) - 34 KB:
Download: *http://www.linuxfromscratch.org/lfs/downloads/7.5/lfs-bootscripts-20130821.tar.bz2*
MD5 sum: `e908023fc44e613ad0c81241781289e7`

• Libpipeline (1.2.6) - 761 KB:
Home page: *http://libpipeline.nongnu.org/*
Download: *http://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.2.6.tar.gz*
MD5 sum: `6d1d51a5dc102af41e0d269d2a31e6f9`

• Libtool (2.4.2) - 2,571 KB:
Home page: *http://www.gnu.org/software/libtool/*
Download: *http://ftp.gnu.org/gnu/libtool/libtool-2.4.2.tar.gz*
MD5 sum: `d2f3b7d4627e69e13514a40e72a24d50`

• Linux (3.13.3) - 75,393 KB:
Home page: *http://www.kernel.org/*
Download: *http://www.kernel.org/pub/linux/kernel/v3.x/linux-3.13.3.tar.xz*
MD5 sum: `ad98a0c623a124a25dab86406ddc7119`

Nota

O kernel Linux é atualizado com relativa frequência, muitas vezes devido a descoberta de vulnerabilidades de segurança. The latest available 3.13.x kernel version should be used, unless the errata page says otherwise.

Para usuários com velocidade limitada ou largura de banda cara que queiram atualizar o kernel Linux, uma versão deste pacote e os patches podem ser baixados separadamente. Isso pode salvar algum tempo ou dinheiro para uma atualização subsequente em nível de patch de uma versão inferior.

• M4 (1.4.17) - 1,122 KB:
Home page: *http://www.gnu.org/software/m4/*
Download: *http://ftp.gnu.org/gnu/m4/m4-1.4.17.tar.xz*
MD5 sum: `12a3c829301a4fd6586a57d3fcf196dc`

• Make (4.0) - 1,311 KB:
Home page: *http://www.gnu.org/software/make/*
Download: *http://ftp.gnu.org/gnu/make/make-4.0.tar.bz2*
MD5 sum: `571d470a7647b455e3af3f92d79f1c18`

• Man-DB (2.6.6) - 1,415 KB:
Home page: *http://www.nongnu.org/man-db/*
Download: *http://download.savannah.gnu.org/releases/man-db/man-db-2.6.6.tar.xz*
MD5 sum: `5d65d66191080c144437a6c854e17868`

• Man-pages (3.59) - 1,172 KB:
Home page: *http://www.kernel.org/doc/man-pages/*
Download: *http://www.kernel.org/pub/linux/docs/man-pages/man-pages-3.59.tar.xz*
MD5 sum: `d8e4d8287a76ee861351b905044c8e92`

• MPC (1.0.2) - 619 KB:
Home page: *http://www.multiprecision.org/*
Download: *http://www.multiprecision.org/mpc/download/mpc-1.0.2.tar.gz*
MD5 sum: `68fadff3358fb3e7976c7a398a0af4c3`

• MPFR (3.1.2) - 1,049 KB:
Home page: *http://www.mpfr.org/*
Download: *http://www.mpfr.org/mpfr-3.1.2/mpfr-3.1.2.tar.xz*
MD5 sum: `e3d203d188b8fe60bb6578dd3152e05c`

• Ncurses (5.9) - 2,760 KB:
Home page: *http://www.gnu.org/software/ncurses/*
Download: *http://ftp.gnu.org/gnu/ncurses/ncurses-5.9.tar.gz*
MD5 sum: `8cb9c412e5f2d96bc6f459aa8c6282a1`

• Patch (2.7.1) - 660 KB:
Home page: *http://savannah.gnu.org/projects/patch/*
Download: *http://ftp.gnu.org/gnu/patch/patch-2.7.1.tar.xz*
MD5 sum: `e9ae5393426d3ad783a300a338c09b72`

• Perl (5.18.2) - 13,730 KB:
Home page: *http://www.perl.org/*
Download: *http://www.cpan.org/src/5.0/perl-5.18.2.tar.bz2*
MD5 sum: `d549b16ee4e9210988da39193a9389c1`

• Pkg-config (0.28) - 1,892 KB:
Home page: *http://www.freedesktop.org/wiki/Software/pkg-config*
Download: *http://pkgconfig.freedesktop.org/releases/pkg-config-0.28.tar.gz*
MD5 sum: `aa3c86e67551adc3ac865160e34a2a0d`

• Procps (3.3.9) - 548 KB:
Home page: *http://sourceforge.net/projects/procps-ng*
Download: *http://sourceforge.net/projects/procps-ng/files/Production/procps-ng-3.3.9.tar.xz*
MD5 sum: `0980646fa25e0be58f7afb6b98f79d74`

• Psmisc (22.20) - 422 KB:
Home page: *http://psmisc.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/psmisc/psmisc-22.20.tar.gz*
MD5 sum: `a25fc99a6dc7fa7ae6e4549be80b401f`

• Readline (6.2) - 2,225 KB:
Home page: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
Download: *http://ftp.gnu.org/gnu/readline/readline-6.2.tar.gz*
MD5 sum: `67948acb2ca081f23359d0256e9a271c`

• Sed (4.2.2) - 1,035 KB:
Home page: *http://www.gnu.org/software/sed/*
Download: *http://ftp.gnu.org/gnu/sed/sed-4.2.2.tar.bz2*
MD5 sum: `7ffe1c7cdc3233e1e0c4b502df253974`

• Shadow (4.1.5.1) - 3,428 KB:
Download: *http://cdn.debian.net/debian/pool/main/s/shadow/shadow_4.1.5.1.orig.tar.gz*
MD5 sum: `ae66de9953f840fb3a97f6148bc39a30`

• Sysklogd (1.5) - 85 KB:
Home page: *http://www.infodrom.org/projects/sysklogd/*
Download: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz*
MD5 sum: `e053094e8103165f98ddafe828f6ae4b`

• Sysvinit (2.88dsf) - 108 KB:
Home page: *http://savannah.nongnu.org/projects/sysvinit*
Download: *http://download.savannah.gnu.org/releases/sysvinit/sysvinit-2.88dsf.tar.bz2*
MD5 sum: `6eda8a97b86e0a6f59dabbf25202aa6f`

• Tar (1.27.1) - 1,835 KB:
Home page: *http://www.gnu.org/software/tar/*
Download: *http://ftp.gnu.org/gnu/tar/tar-1.27.1.tar.xz*
MD5 sum: `e0382a4064e09a4943f3adeff1435978`

• Tcl (8.6.1) - 8,551 KB:
Home page: *http://tcl.sourceforge.net/*
Download: *http://prdownloads.sourceforge.net/tcl/tcl8.6.1-src.tar.gz*
MD5 sum: `aae4b701ee527c6e4e1a6f9c7399882e`

• Time Zone Data (2013i) - 214 KB:
Home page: *http://www.iana.org/time-zones*
Download: *http://www.iana.org/time-zones/repository/releases/tzdata2013i.tar.gz*
MD5 sum: `8bc69eb75bea496ebe1d5a9ab576702d`

• Texinfo (5.2) - 3,724 KB:
Home page: *http://www.gnu.org/software/texinfo/*
Download: *http://ftp.gnu.org/gnu/texinfo/texinfo-5.2.tar.xz*
MD5 sum: `cb489df8a7ee9d10a236197aefdb32c5`

• Systemd (208) - 2,328 KB:
Home page: *http://www.freedesktop.org/wiki/Software/systemd/*
Download: *http://www.freedesktop.org/software/systemd/systemd-208.tar.xz*
MD5 sum: `df64550d92afbffb4f67a434193ee165`

• Udev-lfs Tarball (208) - 29 KB:
Download: *http://anduin.linuxfromscratch.org/sources/other/udev-lfs-208-3.tar.bz2*
MD5 sum: `c0231ff619e567a9b11f912d8a7a404a`

• Util-linux (2.24.1) - 3,461 KB:
Home page: *http://userweb.kernel.org/~kzak/util-linux/*
Download: *http://www.kernel.org/pub/linux/utils/util-linux/v2.24/util-linux-2.24.1.tar.xz*
MD5 sum: `88d46ae23ca599ac5af9cf96b531590f`

• Vim (7.4) - 9,632 KB:
Home page: *http://www.vim.org*
Download: *ftp://ftp.vim.org/pub/vim/unix/vim-7.4.tar.bz2*
MD5 sum: `607e135c559be642f210094ad023dc65`

• Xz Utils (5.0.5) - 894 KB:
Home page: *http://tukaani.org/xz*
Download: *http://tukaani.org/xz/xz-5.0.5.tar.xz*
MD5 sum: `aa17280f4521dbeebed0fbd11cd7fa30`

• Zlib (1.2.8) - 441 KB:
Home page: *http://www.zlib.net/*
Download: *http://www.zlib.net/zlib-1.2.8.tar.xz*
MD5 sum: `28f1205d8dd2001f26fec1e8c2cebe37`

Tamanho total desses pacotes: aproximadamente 322 MB

3.3. Patches Necessários

Em adição aos pacotes, vários patches são necessários. Esses patches corrigem quaisquer erros nos pacotes que deveriam ser consertados por seus mantenedores. Os patches também fazem pequenas modificações para fazer com que os pacotes sejam mais fáceis de se trabalhar. Os seguintes patches serão necessários para construir o sistema LFS:

• Bash Upstream Fixes Patch - 56 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/bash-4.2-fixes-12.patch*
MD5 sum: `419f95c173596aea47a23d922598977a`

• Bzip2 Documentation Patch - 1.6 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/bzip2-1.0.6-install_docs-1.patch*
MD5 sum: `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Internationalization Fixes Patch - 140 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/coreutils-8.22-i18n-4.patch*
MD5 sum: `54c99871cd0ca20f29bdc9462e27f0df`

• Glibc FHS Patch - 2.8 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/glibc-2.19-fhs-1.patch*
MD5 sum: `9a5997c3452909b1769918c759eff8a2`

• Kbd Backspace/Delete Fix Patch - 12 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/kbd-2.0.1-backspace-1.patch*
MD5 sum: `f75cca16a38da6caa7d52151f7136895`

• Perl Libc Patch - 1.6 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/perl-5.18.2-libc-1.patch*
MD5 sum: `daf5c64fd7311e924966842680535f8f`

• Readline Upstream Fixes Patch - 3.3 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/readline-6.2-fixes-2.patch*
MD5 sum: `b793b2bf1306bc62e5f1e7ebbdae2f35`

• Sysvinit Consolidated Patch - 3.9 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/sysvinit-2.88dsf-consolidated-1.patch*
MD5 sum: `0b7b5ea568a878fdcc4057b2bf36e5cb`

• Tar Manpage Patch - 7.8 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/7.5/tar-1.27.1-manpage-1.patch*
MD5 sum: `321f85ec32733b1a9399e788714a5156`

Tamanho total desses patches: aproximadamente 229 KB

Em adição aos patches requeridos acima, há uma variedade de patches criados pela comunidade LFS. Esses patches opcionais resolvem problemas menores ou habilitam funcionalidades que não são habilitadas por padrão. Sinta-se livre para examinar o banco de patches localizado no endereço *http://www.linuxfromscratch.org/patches/downloads/* e obtenha quaisquer patches adicionais que sirvam às necessidades de seu sistema.

Capítulo 4. Preparações Finais

4.1. Sobre $LFS

A variável de ambiente `LFS` será usada durante todo o livro. É imprescindível que esta variável sempre esteja definida. Ela deve ser configurada para indicar o ponto de montagem da partição LFS. Verifique se a variável `LFS` está configurada propriamente com o comando:

```
echo $LFS
```

Certifique-se que a saída mostra o caminho para o ponto de montagem da partição LFS, que é `/mnt/lfs` se o exemplo fornecido foi seguido. Se a saída não estiver correta, a variável pode ser configurada com:

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

Ter essa variável configurada é útil uma vez que comandos como **mkdir $LFS/tools** podem ser digitados exatamente como apresentados no livro. O shell irá substituir automaticamente o "$LFS" por "/mnt/lfs" (ou com o qualquer outro valor atribuído a variável) quando ele processar a linha de comando.

Não esqueça de verificar se a variável `$LFS` está configurada quando você deixar ou retornar ao ambiente de trabalho (quando se executa um **su** para entrar como `root` ou outro usuário).

4.2. Criando o diretório $LFS/tools

Todos os programas compilados no Capítulo 5 serão instalados no diretório `$LFS/tools` para mantê-los separados dos programas compilados no Capítulo 6. Os programas compilados aqui serão ferramentas temporárias e não serão parte do sistema LFS final. Mantendo esses programas em um diretório separado, eles podem facilmente ser descartados após seu uso. Isso também previne que esses programas terminem nos diretórios de produção do sistema anfitrião (fácil de acontecer acidentalmente no Capítulo 5).

Crie os diretórios solicitado executando o comando seguinte como `root`:

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

O próximo passo é criar um link simbólico `/tools` no sistema anfitrião. Esse link vai apontar para o diretório recém-criado na partição LFS. Execute também esse comando como `root`:

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

Nota

O comando acima está correto. O comando **ln** tem algumas variações sintáticas, então certifique-se de checar **info coreutils ln** e `ln(1)` antes de reportar o que você pode achar que seja um erro.

O link simbólico criado habilita as ferramentas utilizadas na construção do sistema a serem compiladas de tal forma que sempre se faz referência a `/tools`, significando que o compilador, assembler e linker funcionarão tanto no Capítulo 5 (quando nós ainda usaremos algumas ferramentas do sistema anfitrião) quanto no seguinte (quando nós mudaremos, "chrooted" , para a partição LFS).

4.3. Adicionando o usuário LFS

Quando logado como usuário `root`, cometer um simples erro pode danificar ou destruir um sistema. Portanto, nós recomendamos construir os pacotes neste capítulo como um usuário sem privilégios. Você poderia usar seu próprio usuário, mas para facilitar a configuração de um ambiente de trabalho limpo, crie um novo usuário chamado `lfs` como um membro de um novo grupo (também chamado `lfs`) e use esse usuário durante o processo de instalação. Como `root`, execute os seguintes comandos para adicionar o novo usuário:

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

O significado das opções do comando:

`-s /bin/bash`
Faz do **bash** o shell padrão para o usuário `lfs`.

`-g lfs`
Adiciona o usuário `lfs` ao grupo `lfs`.

`-m`
Esta opção cria um diretório de usuário (home) para o usuário `lfs`.

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

`lfs`
Esse é o nome real para o grupo e usuário criados.

Para entrar como `lfs` (em oposição a mudar para o usuário `lfs` quando logado como `root`, que não requer que o usuário `lfs` tenha uma senha), dê ao usuário `lfs` uma senha:

```
passwd lfs
```

Garanta ao usuário `lfs` acesso total ao diretório `$LFS/tools` fazendo o usuário `lfs` dono do diretório:

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

Se um diretório de trabalho separado foi criado como sugerido, dê ao usuário `lfs` a posse do mesmo:

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

Em seguida, entre como usuário `lfs`. Isso pode ser feito via terminal virtual, através de um gerenciador de tela, ou com o comando seguinte para mudança de usuário:

```
su - lfs
```

O "`-`" instrui o comando **su** a iniciar um shell de login em vez de um shell non-login. A diferença entre esses dois tipos de shell 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 logado como usuário `lfs`, iexecute o comando seguinte para criar um novo arquivo `.bash_profile`:

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

Enquanto logado como usuário `lfs`, o shell inicial é geralmente o shell de *login* que lê o arquivo `/etc/profile` do sistema anfitrião (que provavelmente contém algumas configurações e variáveis de ambiente) e então lê o arquivo `.bash_profile`. O comando **exec env -i.../bin/bash** no arquivo `.bash_profile` substitui o shell em utilização por um novo com um ambiente completamente vazio, exceto pelas variáveis `HOME`, `TERM`, e `PS1`. Isso garante que nenhuma variável de ambiente indesejável e potencialmente danosa do sistema anfitrião vaze para o listema em construção. A técnica usada aqui faz com que se alcance o objetivo de um ambiente limpo.

A nova instância do shell é um shell *non-login*, que não lê os arquivos `/etc/profile` ou `.bash_profile`, mas lê 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=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
EOF
```

O comando **set +h** desativa a função hash do **bash**. Geralmente "hashing" é uma característica útil—**bash** usa uma tabela hash (tabela de dispersão ou tabela de espalhamento) para lembrar o caminho completo de arquivos executáveis evitando procurar no `PATH` várias vezes pelo mesmo executável. Entretanto, as novas ferramentas devem ser usadas tão logo estejam instaladas. Desativando a função hasg, o shell vai sempre procurar no `PATH` quando um programa estiver para ser executado. Dessa forma, o shell encontrará as ferramentas recém compiladas em `$LFS/tools` tão logo elas estejam disponíveis sem lembrar da versão anterior dos mesmos programas que estavam em uma localização diferente.

Configurando a máscara de criação de arquivos do usuário (umask) para 022 garante que somente o dono dos arquivos e diretórios recém-criados tem poder de escrita, mas eles podem ser lidos e executados por qualquer um (assumindo que os modos padrão são usados pelas chamadas de sistema `open(2)`, novos arquivos ficam com permissão 644 e diretórios com modo 755).

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

A variável `LC_ALL` controla a localização de certos programas, fazendo duas mensagens seguirem as convenções de um país específico. Se o sistema anfitrião usa uma Glibc mais velha que 2.2.4, ter `LC_ALL` configurado para qualquer coisa diferente de "POSIX" ou "C" (durante esse capítulo) pode causar problemas se você deixar o ambiente de trabalho de desejar retornar depois. Configurando `LC_ALL` para "POSIX" ou "C" (as duas são equivalentes) garante-se que tudo vai funcionar como esperado no ambiente de trabalho.

A variável `LFS_TGT` define uma descrição de máquina fora do padrão, mas compatível para uso quando construindo nosso compilador cruzado (cross compiler) e linker e quando usando os mesmos para compilação do conjunto de ferramentas para desenvolvimento (toolchain). Mais informações estão disponíveis na Seção 5.2, "Notas Técnicas sobre Toolchain (ferramentas para desenvolvimento do sistema)".

Colocando `/tools/bin` antes do padrão no `PATH`, todos os programas instalados no Capítulo 5 são encontrados pelo shell imediatamente após sua instalação. sso, combinado com a desativação do hashing, limita o risco de programas velhos do sistema anfitrião serem usados quando os mesmos programas estão disponíveis no ambiente do Capítulo 5.

Finalmente, para ter o ambiente totalmente preparado para construção das ferramentas temporárias, use o comando source no recém-criado aquivo .bash_profile:

```
source ~/.bash_profile
```

4.5. Sobre SBUs

Muitas pessoas gostariam de saber de antemão quanto tempo aproximadamente leva para compilar e instalar cada pacote. Devido ao fato do Linux From Scratch poder ser construído em vários sistemas diferentes, é impossível disponibilizar estimativas de tempo apuradas. O maior pacote (Glibc) levará aproximadamente 20 minutos em sistemas mais rápidos, mas poderia levar até 3 dias em sistemas mais lentos! Em vez de disponibilizar tempos, a unidade de medida padrão de construção SBU (Standard Build Unit) será usada.

A unidade SBU será usada como se segue. O primeiro pacote a ser compilado neste livro é Binutils no Capítulo 5. O tempo necessário para compilar esse pacote é que será referência como SBU. Todos os outros tempos de compilação serão expressos relativamente a esse tempo.

Por exemplo, considere um pacote cujo tempo de compilação é 4.5 SBUs. Isso significa que se um sistema precisou de 10 minutos para compilar e instalar a primeira passagem do Binutils, será necessário *aproximadamente* 45 minutos para construir esse pacote exemplo. Felizmente, a maioria dos tempos de construção são menores do que o tempo para o Binutils.

De maneira geral, SBUs não são totalmente acuradas porque dependem de muitos fatores, incluindo a versão do GCC do sistema anfitrião. Elas são disponibilizadas aqui para dar uma estimativa de quanto tempo pode levar para instalar um pacote, mas os números podem varia por dúzias de minutos em alguns casos.

Para ver o tempo real para algumas máquinas específicas, nós recomendamos The LinuxFromScratch SBU Home Page no endereço *http://www.linuxfromscratch.org/~sbu/*.

Nota

Para muitos sistemas modernos com múltiplos processadores (ou cores) o tempo de compilação para um pacote pode ser reduzido usando um “parallel make”, o que pode ser feito tanto configurando uma variável de ambiente ou dizendo para o programa **make** quantos processadores estão disponíveis. For exemplo, um Core2Duo pode suportar dois processos simultâneos com:

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

ou só construindo com:

```
make -j2
```

Quando múltiplos processadores são usados dessa maneira, as unidades SBU no livro irão variar mais do que normalmente aconteceria. Analisando a saída dos processos de construção também será mais difícil porque as linhas dos diferentes processos estarão intercaladas. Se você tiver problemas com uma passagem durante a construção, retorne para um único processador para analisar devidamente as mensagens de erro.

4.6. Sobre as Ferramentas de Teste (Suites de Teste)

A maioria dos pacotes disponibiliza ferramentas de teste. Rodar as ferramentas de teste para um pacote recém construído é uma boa ideia porque isso pode prover um “sverificação de sanidade” indicando que tudo foi devidamente compilado. Uma ferramenta de testes que executa seu conjunto de checagens geralmente prova que o pacote está funcionando como o desenvolvedor pretendia. Entretanto isso não garante que o pacote está totalmente livre de bugs.

Algumas suites de teste são mais importantes que outras. Por exemplo, as suites de teste para os pacotes principais das ferramentas de desenvolvimento (toolchain)—GCC, Binutils, e Glibc—são de máxima importância devido a seu papel central em um sistema que funcione corretamente. As suites de teste para GCC e Glibc podem levar bastante tempo para completarem, especialmente em uma máquina lenta, mas são fortemente recomendadas.

Nota

Experiência tem mostrado que há pouco a se ganhar executando as suites de teste no Capítulo 5. Não há escapatória ao fato de que o sistema anfitrião sempre exerce alguma influência sobre os testes naquele capítulo, geralmente causando falhas inexplicáveis. Devido ao fato das ferramentas construídas no Capítulo 5 serem temporárias e eventualmente descartadas, nós não recomendamos executar as suites de teste no Capítulo 5 para a maioria dos leitores. As instruções para a execução dessas suites de testes são disponibilizadas para beneficiar os desenvolvedores e analistas (testers), mas elas são opcionais.

Um problema comum enquanto executando as suites 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 diversos motivos, mas a causa mais provável é que o sistema anfitrião não tem os arquivos de sistema `devpts` configurados corretamente. Esse problema é discutido em maiores detalhes na página *http://www.linuxfromscratch.org/lfs/faq.html#no-ptys*.

Algumas vezes suites de testes falham, mas por razões das quais os desenvolvedores estão cientes e não são consideradas críticas. Consulte os registros (logs) localizados em *http://www.linuxfromscratch.org/lfs/build-logs/7.5/* para verificar quando essas falhas são esperadas ou não. Este site é válido para todos os testes durante este livro.

Capítulo 5. Construindo um Sistema Temporário

5.1. Introdução

Este capítulo mostra como construir um sistema Linux mínimo. Este sistema irá conter somente as ferramentas necessárias para iniciar a construção do sistema LFS final no Capítulo 6 e disponibilizará um ambiente de trabalho mais conveniente para o usuário do que um ambiente mínimo.

Há dois passos na construção de um ambiente mínimo. O primeiro passo é construir um novo conjunto de ferramentas de desenvolvimento independente do sistema anfitrião (compilador, assembler, linker, bibliotecas, e algumas ferramentas úteis). O segundo passo usa esse conjunto de ferramentas para construir outras ferramentas essenciais.

Os arquivos compilados neste capítulo serão instalados no diretório `$LFS/tools` para mantê-los separados dos arquivos que serão instalados nos próximos capítulos e das ferramentas do sistema anfitrião. Uma vez que os arquivos aqui compilados são temporários, nós não queremos que eles poluam o sistema LFS que está por vir.

5.2. Notas Técnicas sobre Toolchain (ferramentas para desenvolvimento do sistema)

Esta seção explica um pouco da lógica e dos detalhes técnicos por trás do método de construção como um todo. Não se faz essencial entender imediatamente tudo que está nesta seção. Boa parte destas informações ficarão claras após uma construção. Esta seção pode ser consultada a qualquer tempo durante o processo.

O objetivo geral do Capítulo 5 é produzir uma área temporária que contém um conjunto de ferramentas em cujo funcionamento pode-se confiar e que possa ser isolado do sistema anfitrião. Usando **chroot**, os comandos nos capítulos restantes estarão restritos àquele ambiente, garantindo uma construção limpa e livre de problemas do novo sistema LFS. O processo de construção foi planejado para minimizar os riscos para os leitores novos e ao mesmo tempo prover um maior valor educacional.

Nota

Antes de continuar, esteja ciente do nome da plataforma de trabalho, geralmente referido como “trio alvo” (usaremos o termo original em inglês “target triplet” que é nome do sistema geralmente na forma “cpu-vendor-os”). Uma maneira simples de determinar o target triplet é executar o script **config.guess** sque vem com o código fonte de muitos pacotes. Extraia o pacote Binutils e execute o script: **`./config.guess`** e tome nota da saída. Por exemplo, para um sistema moderno com processador Intel 32-bit a saída será algo parecido com *i686-pc-linux-gnu*.

Também esteja ciente do nome do linker dinâmico da plataforma, geralmente referido como carregador dinâmico (não confundir com o linker padrão **ld** que é parte do Binutils). O linker dinâmico disponibilizado pela Glibc encontra e carrega bibliotecas compartilhadas necessárias para um programa, prepara o programa para rodar, e então executa o programa. O nome do linker dinâmico para uma máquina Inter 32-bit será `ld-linux.so.2`. Uma maneira infalível para determinar o nome de um linker dinâmico é inspecionar um binário aleatório do sistema anfitrião executando o comando: **`readelf -l <nome do binário> | grep interpreter`** e tomar nota da saída. A referência confiável cobrindo todas as plataformas está no arquivo `shlib-versions` na pasta raiz do código fonte do Glibc.

Alguns pontos técnicos chaves sobre como o método de construção do Capítulo 5 funciona:

- Ajustar o nome da plataforma de trabalho, mudando o campo "vendor" do target triplet pela variável `LFS_TGT` garante que a primeira construção do Binutils e GCC produza um cross-linker e cross-compiler compatíveis. Em vez de produzir binários para outra arquitetura, o cross-linker e o cross-compiler produzirão binários compatíveis com o hardware corrente.
- É feita compilação cruzada das bibliotecas temporárias. Devido ao compilador cruzado, por sua natureza, não poder contar com nada do sistema anfitrião, este método remove a possibilidade de uma contaminação do sistema alvo diminuindo a chance dos cabeçalhos ou bibliotecas do sistema anfitrião serem incorporadas nas novas ferramentas. Compilação cruzada também permite que se construam bibliotecas 32-bit e 64-bit em máquinas 64-bit.
- Manipulação cuidadosa dos fontes do GCC diz ao compilador qual linker dinâmico será usado.

Binutils é instalado primeiro porque o script **configure** do GCC e do Glibc executam vários testes no assembler e no linker para determinar que características desses softwares serão habilitadas ou desabilitadas. Isso é mais importante do que se pode perceber. Uma configuração incorreta do GCC ou Glibc pode resultar em uma falha do conjunto de ferramentas, onde o impacto de tal falha pode não aparecer até o final da construção de toda uma distribuição. Uma falha apontada pela suite de testes geralmente irá destacar esse erro antes que mais trabalho seja executado.

Binutils instala seu assembler e linker em duas localizações, `/tools/bin` e `/tools/$LFS_TGT/bin`. As ferramentas em uma localização são ligadas por um hard link em outra. Uma faceta importante do linker é sua ordem de busca de biblioteca. Informações detalhadas podem ser obtidas com o comando **ld** usando a flag *`--verbose`*. Por exemplo, **`ld --verbose | grep SEARCH`** irá apresentar o caminho de busca atual em sua ordem. Ele mostra que arquivos são ligados através do comando **ld** compilado um programa dummy e passando o parâmetro *`--verbose`* spara o linker. Por exemplo, **`gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded`** irá mostrar todos os arquivos abertos com sucesso durante o linking.

O próximo pacote instalado é o GCC. Um exemplo do que pode ser visto durante a execução do seu **configure** é:

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

Isso pe importante pelas razões mencionadas acima. Isso também demonstra que o script configure do GCC não busca que ferramentas usar nos diretórios contidos no PATH. Entretanto, durante a operação de fato do **gcc** os mesmos caminhos de busca não são necessariamente usados. Para descobrir que linker padrão o **gcc** irá usar, execute: **`gcc -print-prog-name=ld`**.

Informações detalhadas podem ser obtidas do **gcc** passando o parâmetro *`-v`* enquanto compila um programa dummy. Por exemplo, **`gcc -v dummy.c`** irá mostrar informações detalhadas sobre o processadr, compilação e estágios de montagem, incluindo os caminhos de busca do **gcc** e sua ordem.

A seguir são instalados os "Linux API headers organizados". Eles permitem que a biblioteca padrão C (Glibc)tenham uma interface com características que o Kernel Linux disponibilizará.

O próximo pacote instalado é Glibc. As considerações mais importantes para construir Glibc são o compilador, binary tools e os cabeçalhos do kernel (kernel headers). O compilador geralmente não é um problema, uma vez que Glibc sempre utilizará o compilador relacionado ao parâmetro *`--host`* passado em seu script configure, e.g. no nosso caso, **i686-lfs-linux-gnu-gcc**. As ferramentas de binários e os cabeçalhos do kernel podem ser um pouco mais complicados. Portanto, não arrisque e use as opções do configure disponíveis para reforçar seleção correta. Após executar **configure**, verifique o conteúdo do arquivo `config.make` no diretório `glibc-build` para checar todos os detalhes importantes. Note o uso de *`CC="i686-lfs-gnu-gcc"`* para controlar quais ferramentas de binários são

usadas e o uso de *-nostdinc* e *-isystem* para controlar os locais de busca do compilador. Esses itens destacam um aspecto importante do pacote Glibc—ele é extremamente auto-suficiente em termos de sua máquina de construção e não depende de maneira geral dos padrões das ferramentas.

Durante o segunda passe do Binutils, nós podemos usar o parâmetro *--with-lib-path* no configure para controlar o caminho de busca do **ld**.

Para o segundo passe do GCC, sua fonte também precisa ser modificada para dizer ao GCC para usar o novo linker dinâmico. Falha ao fazer isso irá resultar nos programas do GCC tendo o nome do linker dinâmico contido no diretório `/lib` do sistema anfitrião embutido neles, o que acaba com o objetivo de se livrar o sistema anfitrião. Deste ponto em diante, o núcleo das ferramentas para desenvolvimento estará auto-contido e auto-abrigado. O restante dos pacotes do Capítulo 5 são todos construídos com a nova Glibc em `/tools`.

Ao entrar no novo ambiente com chroot no Capítulo 6, o primeiro pacote principal a ser instalado é o Glib, devido a sua natureza auto-suficiente mencionada acima. Uma vez que o Glibc esteja instalado em `/usr`, nós executaremos uma rápida mudança nos padrões do toolchain, e então prosseguiremos construindo o resto do sistema LFS.

5.3. Instruções gerais para compilação

Enquanto construindo os pacotes há várias suposições feitas dentro das intruções:

- Vários dos pacotes recebem patches antes da compilação, mas apenas quando o patch é necessário para evitar um problema. Um patch é frequentemente necessário tanto neste capítulo quanto no próximo, mas às vezes em apenas um. Portanto, não se preocupe se as instruções para um patch baixado pareçam estar faltando. Mensagens de "warning" sobre *offset* ou *fuzz* também podem ser encontradas quando aplicando um patch. Não se preocupe com esses "warnings", caso o patch tenha sido aplicado com sucesso.
- Durante a compilação de muitos pacotes, aparecerão vários "warnings" na tela. Esses são normais e podem ser ignorados com segurança. Esses "warnings" são o que parecem—alertas sobre sintaxe C ou C++ obsoleta, mas não inválida. Padrões C mudam com freqüência, e alguns pacotes ainda usam o padrão antigo. Isso não é um problema, mas gera um "warning".
- Verifique uma última vez se a variável `LFS` está configurada adequadamente:

```
echo $LFS
```

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

- Finalmente, dois últimos pontos tem que ser enfatizados:

Importante

As instruções para construção assumem que as Host System Requirements, incluindo links simbólicos, foram configuradas corretamente:

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

Importante

Para re-enfatizar o processo de construção:

1. Coloque todos os pacotes e patches em um diretório que será acessado a partir do ambiente "chroot", tal como `/mnt/lfs/sources/`. *Não* coloque os pacotes no diretório `/mnt/lfs/tools/`.
2. Mude para o diretório dos pacotes.
3. Para cada pacote:
   a. Usando o programa **tar**, extraia o pacote para ser construído. No Capítulo 5, certifique-se que você seja o usuário *lfs* enquanto extrair os pacotes.
   b. Mude para o diretório criado quando o pacote tiver sido extraído.
   c. Siga as instruções do livro para construir o pacote.
   d. Volte para o diretório dos pacotes (source).
   e. Delete o pacote extraído e quaisquer diretórios `<package>-build` que foram criados durante o processo de construção, a não ser que instruído o contrário.

5.4. Binutils-2.24 - Pass 1

O pacote Binutils contém um linker, um assembler, e outras ferramentas para manusear arquivos objeto.

Tempo de Construção: 1 SBU
Espaço de disco: 404 MB

5.4.1. Instalação do Cross Binutils

Nota

Volte e leia novamente as notas na seção anterior. Entender as notas marcadas como importante fará você evitar muitos problemas depois.

É importante que o Binutils seja o primeiro pacote compilado porque ambos Glibc e GCC executam vários testes sobre o linker e o assembler disponíveis para determinar quais de suas próprias funções habilitar.

A documentação do Binutils recomenda construir o Binutils fora do diretório onde está o código fonte (source) em um diretório dedicado a contrução (build)

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Nota

Para que os valores SBU listados no resto do livro tenham alguma utilidade, conte o tempo que leva para construir este pacote desde a configuração até (incluindo) o primeiro install. Para fazer isso facilmente, encapsule os comandos com um comando **time** desta forma: **`time { ./configure ... && ... && make install; }`**.

Nota

Os palores aproximados do SBU e espaço de disco no Capítulo 5 não incluem os dados da suite de testes.

Agora prepare Binutils para a compilação:

```
../binutils-2.24/configure     \
    --prefix=/tools            \
    --with-sysroot=$LFS        \
    --with-lib-path=/tools/lib \
    --target=$LFS_TGT          \
    --disable-nls              \
    --disable-werror
```

O significado das opções do configure:

`--prefix=/tools`
Isso diz para o script de configuração para preparar para instalar o Binutils no diretório `/tools`.

`--with-sysroot=$LFS`
Força a cross compialçao, isso diz ao sistema de construção para procurar em $LFS as bibliotecas de sistema conforme necessário.

`--with-lib-path=/tools/lib`
Isso especifica que caminho para biblioteca o linker deve ser configurado para usar.

`--target=$LFS_TGT`
Devido ao fato da descrição da máquina na variável `LFS_TGT` ser ligeiramente diferente do valor retornado pelo script **config.guess**, esta chave dirá ao script **configure** para ajustar o sistema de construção do Binutils para construir um cross linker.

`--disable-nls`
Isso desabilita internacionalização, uma vez que i18n não é necessário para as ferramentas temporárias.

`--disable-werror`
Isso previne que a construção pare no caso de warnings serem emitidos do compilador do sistema anfitrião.

Continue compilando o pacote:

```
make
```

A Compilação agora está completa. Ordinariamente agora nós executaríamos a suite de testes, mas neste estágio inicial o framework de teste (Tcl, Expect, e DejaGNU) ainda não estão no lugar. Os benefícios de executar os testes neste momento são mínimos uma vez que programas desta primeira passagem logo serão substituídos pelos da segunda.

Se construindo em um sistema x86_64, crie um symlink para garantir a sanidade das ferramentas.

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

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.13.2, “Conteúdo do Binutils.”

5.5. GCC-4.8.2 - Pass 1

O pacote GCC contém o Gnu Compiler Collection, que inclui os compiladores C e C++.

Tempo de Construção: 5.5 SBU
Espaço de disco: 1.4 GB

5.5.1. Instalação do Cross GCC

GCC agora requer os pacotes GMP, MPFR e MPC. Como esses pacotes podem não estar incluídos na sua distribuição, eles serão construídos com o GCC. Descompacte cada pacote dentro do diretório source do GCC e renomeie os diretórios resultantes para que os procedimentos de contrução doGCC possam usá-los automaticamente:

Nota

Há mal-entendidos freqüentes sobre este capítulo. Os procedimentos são os mesmos que todos os outros capítulos explicados anteriormente (Package build instructions). Primeiro extraia o tarball gcc do diretório source e então mude para o diretório criado. Apenas então você deve prosseguir para as instruções abaixo.

```
tar -Jxf ../mpfr-3.1.2.tar.xz
mv -v mpfr-3.1.2 mpfr
tar -Jxf ../gmp-5.1.3.tar.xz
mv -v gmp-5.1.3 gmp
tar -zxf ../mpc-1.0.2.tar.gz
mv -v mpc-1.0.2 mpc
```

O comando seguinte erá mudar a localização do linker dinâmico padrão do GCC para usar aquele instalado em `/tools`. Ele também remove `/usr/include` do path de busca do GCC. Execute:

```
for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

No caso do comando acima parecer difícil de seguir, vamos quebrá-lo um pouco. Primeiro nós encontramos todos os arquivos no diretório `gcc/config` que são nomeados como `linux.h`, `linux64.h` ou `sysv4.h`. Para cada arquivo encontrado, nós o copiamos para um arquivo com o mesmo nome, mas com o sufixo ".orig". Então a primeira expressão do comando sed aciciona "/tools" ao início de cada instância "/lib/ld", "/lib64/ld" ou "/lib32/ld", enquanto o segundo substitui instâncias com "/usr". A seguir nós adicionamos nosso enunciado define que altera o prefixo padrão startfile para o final do arquivo. Note que a barra final "/" em "/tools/lib/" é necessária. Finalmente, nós usamos **touch** para atualizar o timestamp dos arquivos copiados. Quando usado em conjunto com **cp -u**, isso previne mudanças inesperadas nos arquivos originais em caso de o comando ser executado duas vezes.

GCC não detecta proteção de pilha atualmente (stack protection),o que causa problemas para a construção do Glibc-2.19, então conserte isso executando o seguinte comando:

```
sed -i '/k prot/agcc_cv_libc_provides_ssp=yes' gcc/configure
```

A documentação do GCC recomenda construir o GCC fora do diretório source em um diretório build dedicado:

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Prepare o GCC para a compilação:

```
../gcc-4.8.2/configure                             \
    --target=$LFS_TGT                              \
    --prefix=/tools                                \
    --with-sysroot=$LFS                            \
    --with-newlib                                  \
    --without-headers                              \
    --with-local-prefix=/tools                     \
    --with-native-system-header-dir=/tools/include \
    --disable-nls                                  \
    --disable-shared                               \
    --disable-multilib                             \
    --disable-decimal-float                        \
    --disable-threads                              \
    --disable-libatomic                            \
    --disable-libgomp                              \
    --disable-libitm                               \
    --disable-libmudflap                           \
    --disable-libquadmath                          \
    --disable-libsanitizer                         \
    --disable-libssp                               \
    --disable-libstdc++-v3                         \
    --enable-languages=c,c++                       \
    --with-mpfr-include=$(pwd)/../gcc-4.8.2/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

O significado das opções do configure:

`--with-newlib`
Uma vez que uma biblioteca C funcional ainda não está disponível, isso assegura que a constante inhibit_libc esteja definida quando construindo libgcc. isso previne a compilação de qualquer código que requer o suporte do libc.

`--without-headers`
Quando criando o compilador cruzado completo, o GCC requer headers padrões compatíveis com o sistema a ser criado. Para o nosso propósito esses headers não serão necessários. Este parametro previne o GCC de procurar por eles.

`--with-local-prefix=/tools`
O "local prefix" é a localização no sistema onde o GCC irá procurar por arquivos include instalados localmente. O padrão é `/usr/local`. Configurando isso para `/tools` ajuda a manter a localização do hoste `/usr/local` fora do path de busca do GCC.

`--with-native-system-header-dir=/tools/include`
Por padrão o GCC procura em `/usr/include` por headers de sistema. Em conjunto com a chave sysroot, ele traduziria normalmente para `$LFS/usr/include`. Entretanto os headers instalados nas duas próximas seções irão para `$LFS/tools/include`. Esse switch garante que o gcc irá encontrá-los corretamente. No segundo passe do GCC, este mesmo switch irá garantir que nenhum cabeçalho do sistema anfitrião seja encontrado.

`--disable-shared`
Este switch força o GCC a linkar suas bibliotecas internas estaticamente. Nós fazemos isso para evitar possíveis problemas com o sistema anfitrião.

`--disable-decimal-float, --disable-threads, --disable-libatomic, --disable-libgomp, --disable-libitm, --disable-libmudflap, --disable-libquadmath, --disable-libsanitizer, --disable-libssp, --disable-libstdc++-v3`
Esses switches desabilitam o suporte para a extensão de ponto flutuante decimal, threading, libatomic, libgomp, libitm, libmudflap, libquadmath, libsanitizer, libssp e a biblioteca padrão C++ standard respectivamente. Essas características irão falhar na compilação quando construindo um compilador cruzado e não são necessárias para a tarefa de cross compilar a libc temporária.

`--disable-multilib`
Em um sistema x86_64, LFS ainda não tem suporte a configuração multilib. Este switch é inofensivo para x86.

`--enable-languages=c,c++`
Essa opção garante que apenas os compiladores C e C++ sejam construídos. Estas são as únicas linguagens necessárias agora.

`--with-mpfr-*`
Essas opções habilitam o sistema de construção para usar corretamente a cópia do código MPFR inclusa.

Compile GCC executando:

```
make
```

A Compilação agora está completa. Neste ponto, a suite de teste normalmente seria executada, mas, como mencionado anteriormente, o framework da suite de teste ainda não está no lugar. Os benefícios de executar os testes neste momento são mínimos uma vez que programas desta primeira passagem logo serão substituídos.

Instale o pacote:

```
make install
```

Usar *`--disable-shared`* significa que o arquivo `libgcc_eh.a` não é criado nem instalado. O pacote Glibc depende desta biblioteca uma vez que ele usa *`-lgcc_eh`* em seu sistema de construção. Essa dependência pode ser satisfeita criando um symlink `libgcc.a`, uma vez que esse aquivo conterá os objetos normalmente contidos em `libgcc_eh.a`:

```
ln -sv libgcc.a `$LFS_TGT-gcc -print-libgcc-file-name | sed 's/libgcc/&_eh/'`
```

Detalhes deste pacote estão localizados na Seção 6.17.2, “Conteúdo do GCC.”

5.6. Linux-3.13.3 API Headers

O "Linux API Headers" (em linux-3.13.3.tar.xz) xpõe a API do kernel para uso do Glibc.

Tempo de Construção: 0.1 SBU
Espaço de disco: 584 MB

5.6.1. Instalação do Linux API Headers

O kernel Linux precisa expor uma Interface de Programação de Aplicativos (API) para a biblioteca C do sistema (Glibc no LFS) usar. Isso é feito através de vários arquivos de cabeçalho C que vem junto do código fonte do kernel Linux.

Certifique-se de que não haja arquivos obsoletos e dependências de atividades anteriores:

```
make mrproper
```

Agora teste e extraia do fonte os cabeçalhos do kernel visíveis ao usuário. Eles estão localizados em diretório local intermediário e copiados para a localização necessária porque o processo de extração remove quaisquer arquivos no diretório de destino.

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

Detalhes deste pacote estão localizados na Seção 6.7.2, “Conteúdo do Linux API Headers.”

5.7. Glibc-2.19

O pacote Glibc contém a biblioteca C principal. Este pacote prov# as rotinas básicas para alocação de memória, busca em diretórios, abrir e fechar arquivos, ler e escrever arquivos, manuseio de strings, correspondência de padrões, aritmética, e muito mais.

Tempo de Construção: 4.7 SBU
Espaço de disco: 567 MB

5.7.1. Instalação do Glibc

Em alguns casos, particularmente LFS 7.1, os rpc headers não foram instalados adequadamente. Teste para verificar se eles foram instalados no sistema anfitrião e instale se eles não estiverem instalados:

```
if [ ! -r /usr/include/rpc/types.h ]; then
  su -c 'mkdir -pv /usr/include/rpc'
  su -c 'cp -v sunrpc/rpc/*.h /usr/include/rpc'
fi
```

A documentação do Glibc recomenda construir o Glibc fora do diretório onde está o código fonte (source) em um diretório dedicado a contrução (build):

```
mkdir -v ../glibc-build
cd ../glibc-build
```

A seguir, prepare o Glibc para compilação:

```
../glibc-2.19/configure                             \
      --prefix=/tools                               \
      --host=$LFS_TGT                               \
      --build=$(../glibc-2.19/scripts/config.guess) \
      --disable-profile                             \
      --enable-kernel=2.6.32                        \
      --with-headers=/tools/include                 \
      libc_cv_forced_unwind=yes                     \
      libc_cv_ctors_header=yes                      \
      libc_cv_c_cleanup=yes
```

O significado das opções do configure:

`--host=$LFS_TGT, --build=$(../glibc-2.19/scripts/config.guess)`
O efeito combinado desses switchs é que o sistema de construção do Glibc se configura para compilação cruzada, usando um cross-linker and cross-compiler em `/tools`.

`--disable-profile`
Isso cria as bibliotecas sem informação de profiling. Omita essa opção se profiling for necessário nas ferramentas temporárias.

`--enable-kernel=2.6.32`
sso habilita o Glibc a compilar a biblioteca com suporte ao kernel Linux 2.6.32 e posteriores. Workarounds para kernels antigos não estão habilitados.

`--with-headers=/tools/include`

Isso diz ao Glibc para compilar a si mesmo com os cabeçalhos recentemente instalados no diretório tools, de teal modo que ele sabe exatamente que características o kernel tem e pode otimizar-se adequadamente.

`libc_cv_forced_unwind=yes`

O linker instalado na Seção 5.4, “Binutils-2.24 - Pass 1” foi cross-compilado e dessa forma não pode ser usado até que Glibc tenha sido instalado. Isso significa que o teste do script configure que verifica o suporte a opção force-unwind irá falhar, uma vez que ele depende de um linker funcional. A variável libc_cv_forced_unwind=yes é passada para informa ao **configure** que o suporte a force-unwind support está disponível sem que o teste seja executado.

`libc_cv_c_cleanup=yes`

De maneira semelhante, nós passaos libc_cv_c_cleanup=yes para o script **configure** de modo a pular o teste e fazer com que o suporte a cleanup do C seja configurado.

`libc_cv_ctors_header=yes`

De maneira semelhante, nós passaos libc_cv_ctors_header=yes para o script **configure** de modo a pular o teste e fazer com que o suporte ao contructor gcc seja configurado.

Durante este estágio os seguintes alertas podem 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 programa **msgfmt** faltando ou incompatível geralmente é inofensivo. O programa **msgfmt** é parte do pacote Gettext que o sistema anfitrião deve fornecer.

Compile o pacote:

```
make
```

Este pacote contém uma suite de testes, entretanto, ela não pode ser executada neste momento porque nós ainda não temos um compilador C++.

Nota

A suite de teste também precisa das informações de "locale" instaladas para rodar com sucesso. Os dados do locale fornecem ao sistema informações relacionadas a coisas como data, hora, formato de moeda aceitáveis e saída de utilitários do sistema. Se a suite de testes não for executada neste capítulo (conforme recomendado), não há necessidade de instalar os locales agora. Os locales adequados serão instalados no próximo capítulo. Para instalar os locales do Glibc mesmo assim, use as instruções da Seção 6.9, “Glibc-2.19.”

Instale o pacote:

```
make install
```

Cuidado

Neste ponto é imperativo parar e certificar-se de que as funções básicas (compilar e lincar) do novo toolchain estão funcionando como esperado. Para executar o teste de sanidade, rode o seguinte comando:

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

Se tudo estiver funcionando corretamente, não deve haver qualquer erro e a saída do último comando será na forma:

```
[Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

Note que `/tools/lib`, ou `/tools/lib64` fpara máquinas 64-bit aparece como prefixo do linker dinâmico.

Se a saída não for como mostrado acima ou se não há qualquer saída, então há algo errado. Investigue e retrace os paços para encontrar onde o problema está e corrija o mesmo. Este problema tem que ser resolvido antes de continuar.

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

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

Nota

Construir o Binutils na seção após a próxima servirá como uma checagem adicional de que a toolchain foi propriamente construída. Se o Binutils falhar na construção, isso é uma indicação de que alguma coisa deu errado durante a instalação anterior do Binutils, GCC ou Glibc.

Detalhes deste pacote estão localizados na Seção 6.9.4, “Conteúdo do Glibc.”

5.8. Libstdc++-4.8.2

Libstdc++ é a biblioteca C++ padrão. É necessária para a correta operação do compilador g++.

Tempo de Construção: 0.4 SBU
Espaço de disco: 734 MB

5.8.1. Instalação Target Libstdc++

Nota

Libstdc++ é parte do fonte do GCC. Você deve primeiro desempacotar o tarball GCC e mudar para o diretório `gcc-4.8.2`.

Crie um diretório Libstdc++ e entre nele:

```
mkdir -pv ../gcc-build
cd ../gcc-build
```

Prepare Libstdc++ para compilação:

```
../gcc-4.8.2/libstdc++-v3/configure \
    --host=$LFS_TGT                 \
    --prefix=/tools                 \
    --disable-multilib              \
    --disable-shared                \
    --disable-nls                   \
    --disable-libstdcxx-threads     \
    --disable-libstdcxx-pch         \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/4.8.2
```

O significado das opções do configure:

`--host=...`
Indica para que seja usado o cross compilador que nós acabamos de construir em vez do presente em `/usr/bin`.

`--disable-libstdcxx-threads`
Uma vez que nós não construímos a biblioteca C thread, a do C++ também não pode ser construída.

`--disable-libstdcxx-pch`
Isso previne a instalação de arquivos include precompilados, os quais não são necessário neste momento.

`--with-gxx-include-dir=/tools/include/c++/4.8.2`
Esta é a localização onde os arquivos include padrão são procurados pelo compilador C++. Em uma construção normal, esta informação é automaticamente passada para Libstdc++ **configure** a partir do diretório superior. Em nosso caso, essa informação deve ser explicitamente dada.

Compile libstdc++ executando:

```
make
```

Instale a biblioteca:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.17.2, “Conteúdo do GCC.”

5.9. Binutils-2.24 - Pass 2

O pacote Binutils contém um linker, um assembler, e outras ferramentas para manusear arquivos objeto.

Tempo de Construção: 1.1 SBU
Espaço de disco: 417 MB

5.9.1. Instalação do Binutils

Crie um diretório build separado novamente:

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Prepare Binutils para a compilação:

```
CC=$LFS_TGT-gcc                \
AR=$LFS_TGT-ar                 \
RANLIB=$LFS_TGT-ranlib         \
../binutils-2.24/configure     \
    --prefix=/tools            \
    --disable-nls              \
    --with-lib-path=/tools/lib \
    --with-sysroot
```

O significado das novas opções do configure:

`CC=$LFS_TGT-gcc AR=$LFS_TGT-ar RANLIB=$LFS_TGT-ranlib`
Devido ao fato deste ser realmente uma construção nativa do Binutils, configura essas variáveis garante que o sistema de construção usa o compilador cruzado e ferramentas associadas em vez daquelas do sistema anfitrião.

`--with-lib-path=/tools/lib`
Isso diz para o script de configuração especificar o caminho de busca de biblioteca durante a compilação do Binutils, resultando em `/tools/lib` sendo passado para o linker. Isso previne que o linker pesquise por bibliotecas nos diretórios do sistema anfitrião

`--with-sysroot`
A característica sysroot habilita o linker a encontrar objetos compartilhados os quais são necessários por outros objetos compartilhados explicitamente incluídos na linha de comando do linker. Sem isso, alguns pacotes podem não construir de maneira correta em alguns sistemas.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Agora prepare o linker para a fase de “Reajustagem” no próximo capítulo:

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

O significado dos parâmetros do make:

`-C ld clean`

Isso diz para o programa make remover todos os arquivos compilados no subdiretório `ld`.

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

Esta opção reconstrói tudo no subdiretório `ld`. Especificar a variável `LIB_PATH` do Makefile na linha de comando nos permite sobrescrever o valor padrão das ferramentas temporárias e apontá-lo para o path final adequado. O valor dessas variáveis especifica o caminho de busca padrão da biblioteca do linker. Esta preparação é usada no próximo capítulo.

Detalhes deste pacote estão localizados na Seção 6.13.2, “Conteúdo do Binutils.”

5.10. GCC-4.8.2 - Pass 2

O pacote GCC contém o Gnu Compiler Collection, que inclui os compiladores C e C++.

Tempo de Construção: 7.1 SBU
Espaço de disco: 1.8 GB

5.10.1. Instalação do GCC

Nossa primeira construção do GCC instalou um par de cabeçalhos de sistema internos. Normalmente um deles, `limits.h` irá em retorno incluir o cabeçalho do sistema `limits.h`, neste caso, `/tools/include/limits.h`. Entretanto, no momento da primeira construção do gcc o arquivo `/tools/include/limits.h` não existia, então os cabeçalhos internos que o GCC instalou são parciais, arquivos auto-contidos e não incluem as características internas do cabeçalho do sistema. Isso foi adequado para construir a libc temporária, mas essa construção do GCC agora requer o cabeçalho interno completo. Crie uma versão completa do cabeçalho interno usando um comando que é identico a o que o sistema de construção do GCC cria em circunstâncias normais:

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

Para máquinas x86, uma construção do GCC com bootstrap usa a flag `-fomit-frame-pointer`. Uma construção sem bootstrap omite essa flag por padrão, e o objetivo deveria ser produzir um compilador que é exatamente o mesmo caso tivesse executado a construção com bootstrap. Aplique o seguinte comando **sed** para forçar a construção a usar a flag:

```
case `uname -m` in
  i?86) sed -i 's/^T_CFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in ;;
esac
```

Novamente, mude a localização do linker dinâmico padrão do GCC para usar aquele instalado em `/tools`.

```
for file in \
 $(find gcc/config -name linux64.h -o -name linux.h -o -name sysv4.h)
do
  cp -uv $file{,.orig}
  sed -e 's@/lib\(64\)\?\(32\)\?/ld@/tools&@g' \
      -e 's@/usr@/tools@g' $file.orig > $file
  echo '
#undef STANDARD_STARTFILE_PREFIX_1
#undef STANDARD_STARTFILE_PREFIX_2
#define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
#define STANDARD_STARTFILE_PREFIX_2 ""' >> $file
  touch $file.orig
done
```

Assim como na primeira passagem do GCC ele requer os pacotes GMP, MPFR e MPC. Desempacote os tarballs e mova-os para os nomes de diretórios requeridos:

```
tar -Jxf ../mpfr-3.1.2.tar.xz
mv -v mpfr-3.1.2 mpfr
tar -Jxf ../gmp-5.1.3.tar.xz
mv -v gmp-5.1.3 gmp
tar -zxf ../mpc-1.0.2.tar.gz
mv -v mpc-1.0.2 mpc
```

Crie um diretório build separado novamente:

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Antes de começar a construir o GCC, lembre de remover quaisquer variáveis de ambiente que sobrescrevam as flags padrão de otimização.

Agora prepare o GCC para a compilação:

```
CC=$LFS_TGT-gcc                                    \
CXX=$LFS_TGT-g++                                   \
AR=$LFS_TGT-ar                                     \
RANLIB=$LFS_TGT-ranlib                             \
../gcc-4.8.2/configure                             \
    --prefix=/tools                                \
    --with-local-prefix=/tools                     \
    --with-native-system-header-dir=/tools/include \
    --enable-clocale=gnu                           \
    --enable-shared                                \
    --enable-threads=posix                         \
    --enable-__cxa_atexit                          \
    --enable-languages=c,c++                       \
    --disable-libstdcxx-pch                        \
    --disable-multilib                             \
    --disable-bootstrap                            \
    --disable-libgomp                              \
    --with-mpfr-include=$(pwd)/../gcc-4.8.2/mpfr/src \
    --with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

O significado das novas opções do configure:

`--enable-clocale=gnu`

Eta opção garante que o modelo correto de locale para o C++ seja escolhido em quaisquer circunstâncias. Se o script configure encontrar a locale *de_DE* instalada, ele irá selecionar o modelo gnu de locale adequado. Entretanto, se o locale *de_DE* não estiver instalado, há o risco de construir uma Interface Binária de Aplicação (ABI)-bibliotecas C++ incompatíveis devido ao fato do modelo genérico de locale inadequado ter sido escolhido.

`--enable-threads=posix`

Isso habilita o tratamento de exceções do C++ para código multi-threaded.

--enable-__cxa_atexit

Esta opção permite o uso de `__cxa_atexit`, em vez de `atexit`, para registrar destrutores C++ para local statics e objetos globais. Esta opção é essencial para um tratamento de destrutores totalmente de acordo com os padrões. Ela também afeta a ABI C++, e portanto resulta em bibliotecas C++ compartilhadas e programas C++ que são interoperáveis com outras distribuições Linux.

--enable-languages=c,c++

Essa opção garante que ambos os compiladores C e C++ sejam construídos.

--disable-libstdcxx-pch

Não construa o header pré-compilado (PCH) para `libstdc++`. Ele toma muito espaço e nós não vamos usá-lo.

--disable-bootstrap

Para construções nativas do GCC, o padrão é fazer uma construção "bootstrap". Isso não apenas compila o GCC, mas compila várias vezes. Ele usa os programas compilados em um primeiro round para compilar a si mesmo uma segunda vez e então novamente uma terceira vez. A segunda e a terceira iterações são comparadas para certificar que ele pode reproduzir a si mesmo sem falhas. Isso também implica que ele foi compilado corretamente. Entretanto, o método de construção do LFS deveria prover um compilador sólido sem precisar de bootstrap cada vez.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

Um toque final, crie um symlink. Muitos programas e scripts executam **cc** em vez de **gcc**, o qual é usado para mandar programas genéricos e portanto utilizáveis em todos os tipos de sistemas UNIX onde o compilador GNU C nem sempre está instalado. Executando **cc** deixa o administrador do sistema livre para escolher que compilador C instalar:

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

Cuidado

Neste ponto é imperativo parar e certificar-se de que as funções básicas (compilar e lincar) do novo toolchain estão funcionando como esperado. Para executar o teste de sanidade, rode o seguinte comando:

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

Se tudo estiver funcionando corretamente, não deve haver qualquer erro e a saída do último comando será na forma:

```
[Requesting program interpreter: /tools/lib/ld-linux.so.2]
```

Note que `/tools/lib`, ou `/tools/lib64` fpara máquinas 64-bit aparece como prefixo do linker dinâmico.

Se a saída não for como mostrado acima ou se não há qualquer saída, então há algo errado. Investigue e retrace os paços para encontrar onde o problema está e corrija o mesmo. Este problema tem que ser resolvido antes de continuar. Primeiro, execute o teste de sanidade novamente, usando **gcc** em vez de **cc**. Se isso funcionar, então o symlink `/tools/bin/cc` está faltando. Instale o symlink como mostrado acima. A seguir assegure-se que o `PATH` esteja correto. Isso pode ser checado executando **echo $PATH** e verificando que `/tools/bin` está encabeçando a lista. Se o `PATH` estiver errado isso pode significar que você não está logado como `lfs` ou que algo deu errado na Seção 4.4, “Configurando o Ambiente.”

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

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

Detalhes deste pacote estão localizados na Seção 6.17.2, “Conteúdo do GCC.”

5.11. Tcl-8.6.1

O pacote Tcl contém a linguagem "Tool Command Language".

Tempo de Construção: 0.4 SBU
Espaço de disco: 55 MB

5.11.1. Instalação do Tcl

Este pacote e os p#oximos três (Expect, DejaGNU, e Check) são instalados para dar suporte às suites de teste do GCC, Binutils e outros pacotes. Instalar quatro pacotes para testes pode parecer excessivo, mas é muito assegurador, se não essencial, saber que as ferramentas mais importantes estão funcionando adequadamente. Mesmo que as suites de teste não sejam executadas neste capítulo (elas não são obrigatórias), esses pacotes são necessários para executar as suites de teste no Capítulo 6.

Prepare o Tcl para a compilação:

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

Construa o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Tcl, execute o seguinte comando:

```
TZ=UTC make test
```

A suite de testes do Tcl pode apresentar falhas em certas condições do sistema anfitrião que ainda não são totalmente entendidas. Portanto, falhas na suite de testes não são surpresa e não são consideradas críticas. O parâmetro *TZ=UTC* configura o fuso horário para "Coordinated Universal Time" (UTC), também conhecido com "Greenwich Mean Time" (GMT), mas apenas pela duração da execução da suite de testes. Isso garante que os testes do relógio sejam exercidos corretamente. Detalhes sobre a variável de ambiente TZ esão fornecidos no Capítulo 7.

Instale o pacote:

```
make install
```

Altere a permissão de escrita das bibliotecas instaladas de moto que os símbolos de depuração possam ser removidos posteriormente:

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

Instale os headers do Tcl. O próximo pacote, Expect, requer as mesmas para ser construído.

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

Agora faça um linl simbólico necessário:

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

5.11.2. Conteúdo do Tcl

Programas instalados: tclsh (link to tclsh8.6) and tclsh8.6
Biblioteca instalada: libtcl8.6.so, libtclstub8.6.a

Breve descrição

tclsh8.6	A shell de comando do Tcll
tclsh	A link to tclsh8.6
`libtcl8.6.so`	A biblioteca Tcl
`libtclstub8.6.a`	A biblioteca Stub do Tcl

5.12. Expect-5.45

Este pacote contém programas que executam scripts de diálogos com outros programas interativos.

Tempo de Construção: 0.1 SBU
Espaço de disco: 4.4 MB

5.12.1. Instalação do Expect

Primeiro, force o script configure do Expect a usar `/bin/stty` em vez de um `/usr/local/bin/stty` que ele pode achar no sistema anfitrião. Isso irá assegurar que nossa suite de testes permaneça sã para o final da construção do nosso toolchain:

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

Agora prepare Expect para a compilação:

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

O significado das opções do configure:

`--with-tcl=/tools/lib`
Isso garante que o script configure encontre a instalação do Tcl na localização temporária das ferramentas em vez de possivelmente localizar um existente no sistema anfitrião.

`--with-tclinclude=/tools/include`
Isso explicitamente diz ao Expect onde encontrar os headers internos do Tcl. Usar essa opção evita condições onde o **configure** falha porque ele não pode descobrir automaticamente a localização dos headers do Tcl.

Construa o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Expect, execute o seguinte comando:

```
make test
```

Note que a suite de testes do Expect éconhecida por apresentar falhas sob certas condições do sistema anfitrião que não estão sob nosso controle. Portanto, falhas na suite de testes aqui não são surpresa e não são consideradas críticas.

Instale o pacote:

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

O significado do parâmetro do make:

`SCRIPTS=""`
Isso previne a instalação de scripts suplementares do Expect, os quais não são necessários.

5.12.2. Conteúdo do Expect

Programa instalado: expect
Biblioteca instalada: libexpect-5.45.so

Breve descrição

expect	Comunica-se com outros programas interativos de acordo com um script
`libexpect-5.45.so`	Contém funções que permitem ao Expect ser usado como uma extensão Tcl ou ser usado diretamente do C ou C++ (sem Tcl)

5.13. DejaGNU-1.5.1

O pacote DejaGNU contém um framework para testar outros programas.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 4.1 MB

5.13.1. Instalação do DejaGNU

Prepare o DejaGNU para a compilação:

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

Construa e instale o pacote:

```
make install
```

Teste o resultado, execute:

```
make check
```

5.13.2. Conteúdo do DejaGNU

Programa instalado: runtest

Breve descrição

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

5.14. Check-0.9.12

Check é uma framework de teste de unidade para C.

Tempo de Construção: 0.1 SBU
Espaço de disco: 6.9 MB

5.14.1. Instalação do Check

Prepare o Check para a compilação:

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

O significado do parâmetro do configure:

PKG_CONFIG=
Isso diz ao script configure para ignorar quaisquer opções pkg-config que possam fazer com que o sistema tente linkar com as bibliotecas que não estejam no diretório `/tools`.

Construa o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Check, execute o seguinte comando:

```
make check
```

Note que a suite de teste do Check pode levar um tempo relativamente longo (até 4 SBU).

Instale o pacote:

```
make install
```

5.14.2. Conteúdo do Check

Programa instalado: checkmk
Biblioteca instalada: libcheck.{a,so}

Breve descrição

checkmk	Script awk para gerar teste de unidade C para uso com a framework de teste de unidade Check
`libcheck.{a,so}`	Contém funções que permitem que Check seja chamado de um programa teste

5.15. Ncurses-5.9

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

Tempo de Construção: 0.5 SBU
Espaço de disco: 35 MB

5.15.1. Instalação do Ncurses

Prepare o Ncurses para a compilação:

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

O significado das opções do configure:

`--without-ada`
Isso garante que o Ncurses não construa com suporte ao compilador Ada que pode estar presente no sistema anfitrião, mas não estará disponível quando nós entrarmos no ambiente **chroot**.

`--enable-overwrite`
Isso diz ao Ncurses para instalar seus arquivos de cabeçalho em `/tools/include`, em vez de `/tools/include/ncurses`, para garantir que outros pacotes possam encontrar os cabeçalhos do Ncurses com sucesso.

`--enable-widec`
Este parâmetro faz com que as bibliotecas 'wide-character' (e.g., `libncursesw.so.5.9`) sejam compiladas em vez das normais (e.g., `libncurses.so.5.9`). Essas bibliotecas 'wide-character' são úteis tanto em locales multibyte quando em tradicionais, enquanto as bibliotecas normais só funcionam em locales de 8-bits. Bibliotecas normais e wide-character não compatíveis em código, mas não são compatíveis em binários.

Compile o pacote:

```
make
```

Este pacote contém uma suite de testes, entretanto ela só pode ser executada quando o pacote estiver instalado. O teste reside no diretório `test/`. Veja o arquivo `README` naquele diretório para maiores detalhes.

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.21.2, “Conteúdos do Ncurses.”

5.16. Bash-4.2

Este pacote contém o Burn-Again SHell

Tempo de Construção: 0.4 SBU
Espaço de disco: 48 MB

5.16.1. Instalação do Bash

Primeiro, aplique o patch seguinte para consertar vários bugs que foram abordados anteriormente:

```
patch -Np1 -i ../bash-4.2-fixes-12.patch
```

Prepare o Bash para a compilação:

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

O significado das opções do configure:

`--without-bash-malloc`
Esta opção desativa a função de alocação de memória do Bash (`malloc`) a qual é conhecida por causar falhas de segmentação. Desativando esta opção, o Bash utilizará as funções `malloc` do Glibc que são mais estáveis.

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Bash, execute o seguinte comando:

```
make tests
```

Instale o pacote:

```
make install
```

Faça um link para os programas que usam **sh** como shell:

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

Detalhes deste pacote estão localizados na Seção 6.33.2, “Conteúdo do Bash.”

5.17. Bzip2-1.0.6

O pacote Bzip2 contém programas para compressão e descompressão de arquivos. Comprimir arquivos de texto com **bzip2** gera melhores taxas de compressão do que com o tradicional **gzip**.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 5.7 MB

5.17.1. Instalação do Bzip2

O pacote Bzip2 não contém um script **configure**. Compile e teste com:

```
make
```

Instale o pacote:

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

Detalhes deste pacote estão localizados na Seção 6.19.2, “Conteúdo do Bzip2.”

5.18. Coreutils-8.22

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

Tempo de Construção: 0.8 SBU
Espaço de disco: 133 MB

5.18.1. Instalação do Coreutils

Prepare o Coreutils para a compilação:

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

O significado das opções do configure:

`--enable-install-program=hostname`
Isso habilita o binário **hostname** para ser construído e instalado – é desabilitado por padrão, mas é necessário para a suite de teste do Perl.

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Coreutils, execute o seguinte comando:

```
make RUN_EXPENSIVE_TESTS=yes check
```

O parâmetro *RUN_EXPENSIVE_TESTS=yes* diz para a suite de testes executar vários testes adicionais que são considerados relativamente "caros"(em termos de uso da CPU e da memória) em algumas plataformas, mas geralmente não são um problema no Linux.

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.26.2, “Conteúdo do Coreutils.”

5.19. Diffutils-3.3

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

Tempo de Construção: 0.2 SBU
Espaço de disco: 8.5 MB

5.19.1. Instalação do Diffutils

Prepare o Diffutils para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Diffutils, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.41.2, “Conteúdo do Diffutils.”

5.20. File-5.17

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

Tempo de Construção: 0.1 SBU
Espaço de disco: 12.4 MB

5.20.1. Instalação do File

Prepare o File para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do File, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.12.2, “Conteúdo do File.”

5.21. Findutils-4.4.2

O pacote Findutils contém programas para procurar arquivos. Esses programas são disponibilizados para procurar recursivamente dentro de uma árvore de diretórios e para criar, manter e buscar um banco de dados (geralmente mais rápido que o find recursivo, não confiável de o banco de dados não foi atualizado recentemente).

Tempo de Construção: 0.2 SBU
Espaço de disco: 27 MB

5.21.1. Instalação do Findutils

Prepare o Findutils para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Findutils, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.43.2, “Conteúdo do Findutils.”

5.22. Gawk-4.1.0

O pacote Gawkcontém programas para manipular arquivos de texto.

Tempo de Construção: 0.2 SBU
Espaço de disco: 30 MB

5.22.1. Instalação do Gawk

Prepare o Gawk para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Gawk, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.42.2, “Conteúdo do Gawk.”

5.23. Gettext-0.18.3.2

O pacote Gettext contém utilitários para internacionalização e localização. Eles permitem que programas sejam compilados com NLS (Native Language Support), habilitando-os a emitir mensagens na língua nativa do usuário.

Tempo de Construção: 0.6 SBU
Espaço de disco: 119 MB

5.23.1. Instalação do Gettext

Para o nosso conjunto de ferramentas temporário, nós precisamos apenas construir e instalar três programas do Gettext.

Prepare o Gettext para a compilação:

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

O significado da opção do configure:

`EMACS="no"`
Isso evita que o script configure determine onde instalar os arquivos Emacs Lisp, uma vez que sabe-se que o teste para em alguns sistemas.

`--disable-shared`
Nós não precisamos instalar quaisquer das bibliotecas compartilhadas do Gettext neste momento, portanto não há necessidade de construí-las.

Compile o pacote:

```
make -C gnulib-lib
make -C src msgfmt
make -C src msgmerge
make -C src xgettext
```

Como apenas três programas foram compilados, não é possível rodar a suite de teste sem compilar bibliotecas de suporte adicionais do pacote Gettext. Portanto não é recomendado tentar rodar a suite de testes neste estágio.

Instale os programas **msgfmt**, **msgmerge** e **xgettext**:

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

Detalhes deste pacote estão localizados na Seção 6.44.2, “Conteúdo do Gettext.”

5.24. Grep-2.16

O pacote Grep contém programas para procurar dentro de arquivos.

Tempo de Construção: 0.2 SBU
Espaço de disco: 21 MB

5.24.1. Instalação do Grep

Prepare o Grep para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Grep, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.31.2, “Conteúdo do Grep.”

5.25. Gzip-1.6

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

Tempo de Construção: 0.2 SBU
Espaço de disco: 10 MB

5.25.1. Instalação do Gzip

Prepare o Gzip para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Gzip, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.49.2, “Conteúdo do Gzip.”

5.26. M4-1.4.17

O pacote M4 contém um processador de macro.

Tempo de Construção: 0.2 SBU
Espaço de disco: 30 MB

5.26.1. Instalação do M4

Prepare o M4 para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do M4, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.28.2, “Conteúdo do M4.”

5.27. Make-4.0

O pacote Make contém um programa para compilação de pacotes.

Tempo de Construção: 0.1 SBU
Espaço de disco: 11.2 MB

5.27.1. Instalação do Make

Prepare o Make para a compilação:

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

O significado da opção do configure:

`--without-guile`
Isso garante que o Make-4.0 não será linkado a bibliotecas Guile, as quais podem estar presente no sistema anfitrião, mas não estarão disponíveis dentro do ambiente **chroot** no próximo capítulo.

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Make, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.54.2, “Conteúdo do Make.”

5.28. Patch-2.7.1

O pacote Patch contém um programa para modificar ou criar arquivos aplicando um arquivo “patch” tipicamente criado pelo programa **diff**.

Tempo de Construção: 0.1 SBU
Espaço de disco: 3.4 MB

5.28.1. Instalação do Patch

Prepare o Patch para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Patch, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.55.2, “Conteúdo do Patch.”

5.29. Perl-5.18.2

O pacote Perl contém a linguagem "Practical Extraction and Report Language".

Tempo de Construção: 1.6 SBU
Espaço de disco: 235 MB

5.29.1. Instalação do Perl

Primeiro, aplique o seguinte patch para adaptar alguns "paths" da biblioteca C:

```
patch -Np1 -i ../perl-5.18.2-libc-1.patch
```

Prepare o Perl para a compilação:

```
sh Configure -des -Dprefix=/tools
```

Construa o pacote:

```
make
```

Ainda que Perl venha com uma suite de testes, seria melhor esperar até ele ser instalado no próximo capítulo.

Apenas alguns dos utilitários e bibliotecas precisam ser instalados desta vez:

```
cp -v perl cpan/podlators/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.18.2
cp -Rv lib/* /tools/lib/perl5/5.18.2
```

Detalhes deste pacote estão localizados na Seção 6.38.2, “Conteúdo do Perl.”

5.30. Sed-4.2.2

O pacote Sed contém um editor de fluxo.

Tempo de Construção: 0.1 SBU
Espaço de disco: 10.5 MB

5.30.1. Instalação do Sed

Prepare o Sed para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Sed, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.18.2, “Conteúdo do Sed.”

5.31. Tar-1.27.1

O pacote Tar contém um programa para arquivamento.

Tempo de Construção: 0.4 SBU
Espaço de disco: 20.6 MB

5.31.1. Instalação do Tar

Prepare o Tar para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Tar, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.58.2, “Conteúdo do Tar.”

5.32. Texinfo-5.2

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

Tempo de Construção: 0.3 SBU
Espaço de disco: 94 MB

5.32.1. Instalação do Texinfo

Prepare o Texinfo para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Texinfo, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.59.2, “Conteúdo do Texinfo.”

5.33. Util-linux-2.24.1

O pacote Util-linux contém uma variedade de aplicativos utilitários.

Tempo de Construção: 0.6 SBU
Espaço de disco: 89 MB

5.33.1. Instalação do Util-linux

Prepare Util-linux para a compilação:

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

O significado da opção do configure:

`--disable-makeinstall-chown`
Essa chave desabilita a utilização do comando **chown** durante a instalação. Isso não é necessário quando instalando no diretório /tools e evita a necessidade de se instalar como root.

`--without-systemdsystemunitdir`
Em sistemas que usam systemd, a árvore de pacotes tenta instalar um arquivo systemd específico para um diretório não-existente em /tools. Essa chave desabilita a ação desnecessária.

`PKG_CONFIG=""`
Definindo esta variável de ambiente previne a adição de características desnecessárias que podem estar disponíveis no sistema anfitrião. Note que a localização mostrada para esta variável de ambiente é diferente de outras seções do LFS onde variáveis são definidas anteriormente ao comando. Essa localização é mostrada para demonstrar uma maneira alternativa de definir variáveis de ambiente quando usando configure.

Compile o pacote:

```
make
```

Instale o pacote:

```
make install
```

5.34. Xz-5.0.5

O pacote Xz contém programas para compressão e descompressão de arquivos. Ele fornece recursos para lidar com os novos formatos de compressão lzma e xz. Comprimir arquivos de texto com **xz** gera uma texa de compressão melhor do que os tradicionais comandos **gzip** ou **bzip2**.

Tempo de Construção: 0.2 SBU
Espaço de disco: 16.3 MB

5.34.1. Instalação do Xz

Prepare o Xz para a compilação:

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

Compile o pacote:

```
make
```

A Compilação agora está completa. Como discutido anteriormente, executar a suite de teste não é obrigatório para as ferramentas temporárias aqui neste capítulo. Para executar a suite de teste do Xz, execute o seguinte comando:

```
make check
```

Instale o pacote:

```
make install
```

Detalhes deste pacote estão localizados na Seção 6.46.2, “Conteúdo do Xz.”

5.35. Stripping (Esvaziando)

Os passos nesta seção são opcionais, mas se a partição LFS for pequena, é interessante saber que itens desnecessários podem ser removidos. Os executáveis e bibliotecas construídos até agora contém cerca de 70MB de símbolos de depuração desnecessários. Remova esses símbolos com:

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

Esses comandos vão pular vários arquivos, reportando que não reconhece seu formato de arquivo. Muitos desses são scripts em vez de binários. Use também o comando strip do sistema para incluir o binário strip em /tools.

Take care *not* to use `--strip-unneeded` on the libraries. As estáticas seriam destruídas e todos os pacotes do toolchain precisariam ser construídos novamente.

Para salvar mais, remova a documentação:

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

Neste ponto, voc# deve ter pelo menos 3 GB de espaço livre em `$LFS` que pode ser usado para construir e instalar Glibc na próxima fase. Se você pode construir e instalar Glibc, você pode construir e instalar o resto também.

5.36. Mudando de Posse (Propriedade)

Nota

Os comandos no resto deste livro devem ser executados enquanto logado como `root` e não mais como usuário `lfs`. Além disso, verifique se a variável `$LFS` está configurada no ambiente do `root`.

Atualmente o diretório `$LFS/tools` pertence ao usuário `lfs`, um usuário que existe somente no sistema anfitrião.. Se o diretório `$LFS/tools` for mantido como está, os arquivos pertencem a um ID de usuário sem uma conta. Isso é perigoso porque uma conta de usuário criada posteriormente pode receber esse mesmo ID de usuário e tomar posse do diretório `$LFS/tools` e de todos os arquivos lá contidos, desta forma expondo esses arquivos a uma possível manipulação maldosa.

Para evitar esse problema você poderia adicionar o usuário `lfs` posteriormente ao novo sistema LFS quando criando o arquivo `/etc/passwd` tomando cuidado para atribuir o mesmo usuário e ID de grupo do sistema anfitrião. Ainda melhor, mude o dono do diretório `$LFS/tools` para o usuário `root` executando o seguinte comando:

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

Ainda que o diretóriopossa ser deletado quando o sistema LFS estiver pronto, ele pode ser mantido para construir sistemas LFS adicionais baseados *na mesma versão do livro*. A melhor maneira de fazer backup da pasta `$LFS/tools` é uma questão de preferência pessoal.

Cuidado

Se você pretende manter as ferramentas temporárias para construir sistemas LFS futuramente, *agora* é o momento de fazer backup dessas ferramentas. Comandos subsequentes no capítulo 6 irão alterar as ferramentas atualmente no local, tornando-as inúteis para construções futuras.

Parte III. Construindo o Sistema LFS

Capítulo 6. Instalando os programas do Sistema básico

6.1. Introdução

Neste capítulo, nós entramos na área de construção e começamos a montar o sistema LFS pra valer. Isto é, nós vamos usar o chroot para entrar no mini sistema Linux provisório, vamos fazer algumas preparações finais e vamos então começar a instalar os pacotes.

A instalação destes softwares é precisa. Embora em muitos casos as instruções de instalação pudessem ser mais curtas e genéricas, nós optamos por fornecer as instruções completas para cada pacote a fim de 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 para que você (ou o sistema) pode precisar dele.

Nós não recomendamos usar otimizações. Elas podem fazer um programa rodar ligeiramente mais rápido, mas elas também podem causar dificuldades na compilação e problema quando rodar o programa. Se um pacote recusar a compilar quando usando uma otimização, tente compilar sem a otimização e veja se isso conserta o problema. Mesmo que o pacote compile quando usando otimização, há o risco que ele tenha sido compilado incorretamente devido a complexa interação entre o código e as ferramentas de construção. Note também que as opções `-march` e `-mtune` usando valores não especificados no livro não foram testadas. Isso pode causar problemas com os pacotes do toolchain (Binutils, GCC e Glibc). Os pequenos ganhos potenciais alcançados usando otimizações durante a compilação são geralmente superados pelos riscos. Aqueles que constroem o LFS pela primeira vez são encorajados a não usar otimizações. O sistema subseqüente ainda será rápido e estável ao mesmo tempo.

A ordem em que os pacotes são instalados neste capítulo precisa ser seguida estritamente para garantir que nenhum programa adquira acidentalmente um path que se refere a `/tools`. Pela mesma razão, não compile pacotes separados em paralelo. Compilar em paralelo pode salvar tempo (especialmente em máquias com dual-CPU), mas isso pode resultar em um programa com um path permanente indicando para `/tools`, o que vai fazer com que este programa pare de funcionar quando este diretório for removido.

Antes das instruções de instalação, cada página de instalação provê informações sobre pacotes, incluindo uma descrição concisa do que ele contém, aproximadamente quando tempo leva para compilar, e quanto espaço de disco é necessário durante o processo de construção. Seguindo as instruções de instalação, há uma lista de programas e bibliotecas (juntamente com breves descrições) que o pacote instala.

Nota

Os valores de SBU e de espaço de disco necessário inclui a suite de testes para todos os pacotes aplicáveis no Capítulo 6.

6.2. Preparando sistemas de arquivo virtual do kernel

Vários sistemas de arquivos exportados pelo kernel são usados para comunicação de e para o próprio kernel. Esses sistemas de arquivos são virtuais uma vez que nenhum espaço de disco é utilizado por eles. O conteúdo dos sistemas de arquivos reside na memória.

Comece criando diretórios dentro dos quais os sistemas de arquivos serão montados:

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

6.2.1. Criando nós de dispositivos iniciais

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

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

6.2.2. Montando e Povoando /dev

O método recomendado para povoar o diretório `/dev` com dispositivos é montar um sistema de arquivos virtual (tal como `tmpfs`) no diretório `/dev`, e permitir que os dispositivos sejam criados dinamicamente naquele sistema de arquivos virtual conforme eles sejam detectados ou acessados. Criação de dispositivos é geralmente feita durante o processo de inicialização pelo Udev. Uma vez que este 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 comando seguinte para conseguir isso:

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

6.2.3. Montando os Sistemas de Arquivos Virtuais do Kernel

Agora monte os sistemas de arquivos virtuais do kernel restantes:

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

O significado das opções de montagem para devpts:

`gid=5`

Isso garante que todos os nós de dispositivos criados pelo devpts pertençam ao grudo de ID 5. Este é o ID que nós usaremos posteriormente para o grupo `tty`. Nós usamos o ID de grupo em vez do nome, uma vez que o sistema anfitrião pode usar um ID diferente para seu grupo `tty`.

`mode=0620`

Isso garante que todos os nós de dispositivos criados pelo devpts tenham modo 0620 (escrita e leitura para usuário, escrita para grupo). Juntamente com a opção acima, isso garante que o devpts criará nós de dispositivos que satisfaçam os requisitos do grantpt(), significando que o binário de ajuda do Glibc **pt_chown** (que não é instalado por padrão) não seja necessário.

Em alguns sistemas, `/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
```

6.3. Gerenciamento de Pacote

Gerenciamento de pacotes é uma adição frequentemente solicitada ao livro LFS. Um gerenciador de pacotes permite seguir a instalação de arquivos tornando fácil remover ou atualizar pacotes. Assim como os arquivos binários e bibliotecas, um gerenciador de pacotes irá manipular a instalação de arquivos de configuração. Antes que você comece a perguntar, NÃO—esta seção não falará ou recomendará qualquer gerenciador de pacotes. O que ela disponibiliza é um resumo sobre as técnicas mais populares e como elas funcionam. O gerenciador de pacotes perfeito para você pode estar entre essas técnicas ou pode ser uma combinação de duas ou mais delas. Esta seção menciona brevemente questões que podem aparecer quando atualizando pacotes.

Algumas razões porque nenhum gerenciador de pacotes é mencionado no LFS ou BLFS incluem:

- Lidar com gerenciadores de pacotes retira o foco do objetivo principal do livro—ensinar como um sistema Linux é construído.
- Há múltiplas soluções para gerenciamento de pacotes, cada uma tendo seus pontos fortes e fracos. incluir uma que satisfaça todos os públicos é difícil.

Há algumas dicas escritas no tópico sobre gerenciamento de pacotes. Visite *Hints Project* ae veja se um deles se adequa a suas necessidades.

6.3.1. Problemas de Atualização

Um gerenciador de pacotes ajuda a atualizar para versões novas quando elas são liberadas. Geralmente as instruções nos livros LFS e BLFS podem ser usadas para atualizar para novas versões. Aqui estão alguns pontos que você deve ter em mente quando atualizando pacotes, especialmente em um sistema em execução.

- Se algum dos pacotes do toolchain (Glibc, GCC ou Binutils) precisa ser atualizado para uma nova versão menor, é mais seguro reconstruir o LFS.. Ainda que você *possa* ser capaz de fazê-lo compilando todos os pacotes em sua ordem de dependência, nós não recomendamos isso. Por exemplo, se glibc-2.2.x precisa ser atualizado para glibc-2.3.x, é mais seguro reconstruir. Para atualizações de versão micro, uma simples reinstalação geralmente funciona, mas não é garantido. Por exemplo, atualizar de glibc-2.3.4 para glibc-2.3.5 geralmente não causará quaisquer problemas.
- Se um pacote contendo uma biblioteca compartilhada for atualizado, e se o nome da biblioteca muda, então todos os pacotes dinamicamente vinculados a biblioteca precisam ser recompilados para vincular contra a nova biblioteca. Note que não há qualquer correlação entre a versão do pacote e o nome da biblioteca.) Por exemplo, considere o pacote foo-1.2.3 que instala uma biblioteca compartilhada com o nome `libfoo.so.1`. Digamos que você atualize o pacote para uma nova versão foo-1.2.4 que instala uma biblioteca compartilhada com o nome `libfoo.so.2`. Neste caso, todos os pacotes que são dinamicamente ligados a `libfoo.so.1` precisam ser recompilados para vincular contra `libfoo.so.2`. Note que você não deve remover as bibliotecas anteriores até que os pacotes dependentes sejam recompilados.

6.3.2. Técnicas de Gerenciamento de Pacotes

As seguintes são algumas técnicas de gerenciamento de pacotes comuns. Antes de se decidir com relação a um gerenciador de pacotes, pesquise sobre as várias técnicas, particularmente os pontos fracos do esquema em particular.

6.3.2.1. Está Tudo na Minha Cabeça!

Sim, isto é uma técnica de gerenciamento de pacotes. Algumas pessoas não vêem necessidade em um gerenciador de pacotes porque eles conhecem os pacotes tão intimamente e sabem que arquivos são instalados por cada pacote. Alguns usuários também não precisam de qualquer gerenciamento porque eles planejam reconstruir todo o sistema quando um pacote mudar.

6.3.2.2. Instalação em Diretórios Separados

Este é um gerenciamento de pacotes 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 está instalado em `/usr/pkg/foo-1.1` e um symlink é feito de `/usr/pkg/foo` para `/usr/pkg/foo-1.1`. Quando instalando uma nova versão foo-1.2, ela é instalada em `/usr/pkg/foo-1.2` e o symlink anterior é substituído por um symlink 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 do que alguns poucos pacotes, este sistema se torna impossível de gerenciar.

6.3.2.3. Gerenciamento de Pacotes no Estilo Symlink

Esta é uma variação da técnica de gerenciamento de pacotes anterior. Cada pacote é instalado similar ao esquema anterior. Mas em vez de fazer o symlink, cada arquivo é simbolicamente ligado a hierarquia `/usr`. Isso remove a necessidade de expandir as variáveis de ambiente. Ainda que os symlinks possam ser criados pelo usuário para automatizar a criação, muitos gerenciadores de pacotes tem sido escritos usando esta abordagem. Alguns dos populares inclui Stow, Epkg, Graft, e Depot.

A instalação precisa ser falsa, de modo que o pacote pense que está instalado em `/usr` ainda que ele esteja instalado na hierarquia `/usr/pkg`. Instalação deste modo não é geralmente uma tarefa trivial. Por exemplo, considere que você está instalando um pacote libfoo-1.1. As instruções a seguir podem não instalar o pacote propriamente:

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

A instalação funcionará, mas os pacotes dependentes podem não vincular à libfoo conforme você espera. e você compilar um pacote que vincula à libfoo, você pode notar que ele está linkado para `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` em vez de `/usr/lib/libfoo.so.1` como você esperava. A abordagem correta pe usar a estratégia `DESTDIR` spara fingir 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 esta abordagem, mas há alguns que não. Para os pacotes em não-conformidade, você pode precisar instalar o pacote manualmente, ou você pode perceber que é mais fácil instalar alguns pacotes problemáticos em `/opt`.

6.3.2.4. Baseado em Timestamp (marca temporal)

Nesta técnica, um arquivo recebe uma marca temporal antes da instalação do pacote. Após a instalação, um simples uso do comando **find** com as opções apropriadas pode gerar um log de todos os arquivos instalados após a criação do arquivo com marca temporal. Um gerenciador de pacotes construído com essa abordagem é um "install-log" (registro de instalação).

Ainda que este esquema tenha a vantagem de ser simples, ele tem duas desvantagens. Caso, durante a instalação, os arquivos sejam instalados com qualquer marcação de tempo diferente da hora atual, aqueles arquivos não serão rastreados pelo gerenciador de pacotes. Além disso, este esquema pode ser usado apenas quando um pacote é instalado de cada vez. Os registros não são confiáveis se dois pacotes são instalados em dois consoles diferentes.

6.3.2.5. Script de Rastreamento de Instalação

Nesta abordagem, os comandos que os scripts de instalação executam são gravados. Há 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, esta 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 esta abordagem funcione, todos os executáveis precisam ser dinamicamente vinculados sem o bit suid ou sgid. Pré-carregar a biblioteca pode causar alguns efeitos colaterais indesejados durante a instalação. Portanto, aconselha-se que se execute alguns testes para garantir que o gerenciador de pacotes 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.

6.3.2.6. Criando Arquivos de Pacotes

Neste esquema, a instalação do pacote é simulada em uma árvore de diretório separada como descrito no gerenciamento de pacotes no estilo Symlink. Após instalação, um arquivo do pacote é criado usando os arquivos instalados. Este arquivo é usado para instalar o pacote tanto na máquina local quanto em outras máquinas.

Esta abordagem é usada pela maioria dos gerenciadores de pacotes encontrados em distribuições comerciais. Exemplos de gerenciadores de pacotes que seguem essa abordagem são RPM (o qual, acidentalmente, é necessário para *Linux Standard Base Specification*), pkg-utils, apt do Debian, e sistema Portage do Gentoo. ma dica descrevendo como adotar este estilo de gerenciamento de pacotes para o sistema LFS está localizada em *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

Criação de arquivos pacotes que incluem informações sobre dependências é uma coisa complexa e além do escopo do LFS.

Slackware usa um sistema baseado em **tar** para arquivos de pacotes. Este sistema intencionalmente não manuseia dependências de pacotes como gerenciadores de pacotes complexos fazem. Para detalhes sobre gerenciamento de pacotes no Slackware, veja *http://www.slackbook.org/html/package-management.html*.

6.3.2.7. Gerenciamento Baseado em Usuário

Este sistema, único para LFS, foi desenvolvido por Matthias Benkmann, e está disponível em *Hints Project*. Neste esquema, cada pacote é instalado como um usuário separado nas localizações padrão. Arquivos que pertencem a um usuário são facilmente reconhecidos checando o ID de usuário. As características e desvantagens a esta abordagem são muito complexas para serem descritas nesta seção. Para detalhes, por favor veja a dica em *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

6.3.3. Implantando LFS em Múltiplos Sistemas

Uma das vantagens de um sistema LFS é que não há arquivos que dependam da posição de arquivos em um disco. Clonar uma compilação do LFS para outro computador com uma arquitetura similar a do sistema base é tão simples quanto usar **tar** na partição LFS que contém o diretório raiz (cerca de 250MB descomprimido para um LFS básico), copiando aquele arquivo via rede ou CD-ROM para o novo sistema e expandindo-o. A partir deste ponto, alguns poucos arquivos de configuração terão que ser modificados. Arquivos de configuração que podem precisar de atualização 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 customizado pode ser necessário para o novo sistema dependendo das diferenças entre o hardware dos sistemas e a configuração original do kernel.

Finalmente, o novo sistema deve ser tornado inicializável via Seção 8.4, “Usando o GRUB para Configurar o Processo de Boot”.

6.4. Entrando no ambiente Chroot

É hora de entrar no ambiente chroot para começar a construir e instalar o sistema LFS final. Como usuário `root`, rode o seguinte comando para entrar na área que é, neste momento, populada apenas com as ferramentas temporárias:

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

A opção *`-i`* passada para o comando **env** irá limpar todas as variáveis de ambiente do ambiente chroot. Depois disso, apensa as variáveis `HOME`, `TERM`, `PS1`, e `PATH` são configuradas novamente. A instrução *`TERM=$TERM`* irá configurar a variável `TERM` dentro do ambiente chroot para que tenha o mesmo valor fora deste ambiente. Essa variável é necessária para que programas como **vim** e **less** operem adequadamente. Se outras variáveis forem necessárias, como `CFLAGS` ou `CXXFLAGS`, este é um bom momento para configurá-las novamente.

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

Note que `/tools/bin` vem por último no `PATH`. Isso significa que uma ferramenta temporária não será mais usada uma vez que sua versão final está instalada. Isso acontece porque o shell não “lembra” das localizações dos binários executados—por esta razão, hashing é desabilitado passando a opção *`+h`* para **bash**.

Note que o **bash** irá dizer `I have no name!` sso é normal porque o arquivo `/etc/passwd` fainda 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 sair deste ambiente por qualquer motivo (reiniciar a máquina, por exemplo), certifique-se que o sistema de arquivos virtual do kernel seja montado como explicado em Seção 6.2.2, “Montando e Povoando /dev” e Seção 6.2.3, “Montando os Sistemas de Arquivos Virtuais do Kernel” e entre com o comando chroot novamente antes de continuar a instalação.

6.5. Criando Diretórios

É hora de criar uma estrutura no sistema de arquivos LFS Crie uma árvore de diretórios padrão rodando os seguintes comandos:

```
mkdir -pv /{bin,boot,etc/{opt,sysconfig},home,lib,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -v  /usr/libexec
mkdir -pv /usr/{,local/}share/man/man{1..8}

case $(uname -m) in
 x86_64) ln -sv lib /lib64     &&
         ln -sv lib /usr/lib64 &&
         ln -sv lib /usr/local/lib64 ;;
esac

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

Diretórios são, por padrão, criado 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 do usuário `root`, e outra para os diretórios dos arquivos temporários.

A primeira mudança de modo impede qualquer um de entrar no diretório `/root` —o mesmo que um usuário normal faria com seu diretório home. A segunda mudança de modo garante que qualquer usuário possa escrever nos diretórios `/tmp` e `/var/tmp`, mas não possa remover os arquivos de outros usuários. Esta última é proibida pelo assim chamado "sticky bit," o bit mais alto (1) na máscara 1777.

6.5.1. Nota sobre conformidade com o padrão FHS

A árvore de diretório adotada é baseada no Filesystem Hierarchy Standard (FHS) (disponível em *http://www.pathname.com/fhs/*). Em adição ao FHS, nós criamos symlinks de compatibilidade para os diretórios `man`, `doc`, e `info` uma vez que muitos pacotes ainda tentam instalar sua documentação dentro de `/usr/<directory>` ou `/usr/local/<directory>` em oposição a `/usr/share/<directory>` ou `/usr/local/share/<directory>`. O FHS também estipula a existência de `/usr/local/games` e `/usr/share/games`. O FHS não é preciso quanto a estrutura do subdiretório `/usr/local/share` então nós criamos apenas os diretórios que são necessários. Entretanto, sinta-se livre para criar esses diretórios se você preferir ficar em maior conformidade com o FHS.

6.6. Criando Arquivos e Symlinks Essenciais

Alguns programas usam caminhos para programas que ainda não existem. Para satisfazer esses programas, crie os links simbólicos que serão substituídos pelos arquivos reais no curso deste capítulo após o software ser instalado:

```
ln -sv /tools/bin/{bash,cat,echo,pwd,stty} /bin
ln -sv /tools/bin/perl /usr/bin
ln -sv /tools/lib/libgcc_s.so{,.1} /usr/lib
ln -sv /tools/lib/libstdc++.so{,.6} /usr/lib
sed 's/tools/usr/' /tools/lib/libstdc++.la > /usr/lib/libstdc++.la
ln -sv bash /bin/sh
```

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 o usuário através do sistema de arquivos `/proc`. Para satisfazer utilitários que esperam a presença de `/etc/mtab`, crie o link simbólico:

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

Para que o usuário `root` seja capaz de logar e para que o nome “root” seja reconhecido, deve haver entradas relevantes nos arquivos `/etc/passwd` e `/etc/group`.

Crie o arquivo `/etc/passwd` rodando o seguinte comando::

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

A senha real para o usuário `root` (o “x” usado aqui é só um marcador de espaço) será determinada posteriormente.

Crie o arquivo `/etc/group` odando o seguinte comando:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:
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:
mail:x:34:
nogroup:x:99:
EOF
```

Os grupos criados não são parte de qualquer padrão—eles são grupos decididos em parte pelos requerimento da configuração do Udev neste capítulo, e em parte pelas convenções comuns empregadas em várias distribuições Linux existentes. O Linux Standard Base (LSB, disponível em *http://www.linuxbase.org*) recomenda que, além do grupo `root` com ID (GID) 0, um grupo `bin` com GID 1 esteja presente. Todos os outros nomes de grupos e GIDS podem ser escolhidos livremente pelo administrador do sistema uma vez que programas bem escritos não depende de número de GID, mas sim do nome dos grupos.

Para remover o prompt “I have no name!” inicie um novo shell. Uma vez que o Glibc foi instalado em Capítulo 5 e os arquivos `/etc/passwd` e `/etc/group` foram criados, resolução de usuário e grupo irá funcionar agora:

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

Note o uso da diretiva *+h*. Isso diz ao **bash** nnão usar seu path hashing interno. Sem essa diretiva, **bash** lembraria o caminho de binários que ele já executou. Para garantir o uso de binários recém compilados tão logo eles sejam instalados, a diretiva *+h* será usada pela duração deste capítulo.

Os programas **login**, **agetty**, e **init** (e outros) usam vários arquivos de log para gravar informações como quem esteve logado no sistema e quando. Entretanto, esses programas não irão escrever nos arquivos de log se eles não existirem previamente. Inicialize os arquivos de log e dê a eles a permissão adequada:

```
touch /var/log/{btmp,lastlog,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 usuário logou pela última vez. O arquivo `/var/log/btmp` registra as tentativas de login que falharam.

Nota

O arquivo `/run/utmp` registra quando cada usuário logou pela última vez. Este arquivo é criado dinamicamente nos scripts de boot.

6.7. Linux-3.13.3 API Headers

O "Linux API Headers" (em linux-3.13.3.tar.xz) xpõe a API do kernel para uso do Glibc.

Tempo de Construção: 0.1 SBU
Espaço de disco: 588 MB

6.7.1. Instalação do Linux API Headers

O kernel Linux precisa expor uma Interface de Programação de Aplicativos (API) para a biblioteca C do sistema (Glibc no LFS) usar. Isso é feito através de vários arquivos de cabeçalho C que vem junto do código fonte do kernel Linux.

Certifique-se de que não hava arquivos obsoletos e dependências de atividades anteriores:

```
make mrproper
```

Agora teste e extraia do fonte os cabeçalhos do kernel visíveis ao usuário. Eles estão localizados em diretório local intermediário e copiados para a localização necessária porque o processo de extração remove quaisquer arquivos no diretório de destino. Há também alguns arquivos escondidos usados pelos desenvolvedores do kernel e não necessários para o LFS que são removidos do diretório intermediário.

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

6.7.2. Conteúdo do Linux API Headers

Headers instalados: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h, /usr/include/xen/*.h

Diretórios instalado: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video, /usr/include/xen

Breve descrição

`/usr/include/asm/*.h`	Os cabeçalhos da API de ASM do Linux
`/usr/include/asm-generic/*.h`	Os cabeçalhos da API genérica de ASM do Linux
`/usr/include/drm/*.h`	Os cabeçalhos da API de DRM do Linux
`/usr/include/linux/*.h`	Os cabeçalhos da API do Linux
`/usr/include/mtd/*.h`	Os cabeçalhos da API de MTD do Linux
`/usr/include/rdma/*.h`	Os cabeçalhos da API de RDMA do Linux
`/usr/include/scsi/*.h`	Os cabeçalhos da API de SCSI do Linux
`/usr/include/sound/*.h`	Os cabeçalhos da API de Som do Linux
`/usr/include/video/*.h`	Os cabeçalhos da API de Video do Linux
`/usr/include/xen/*.h`	Os cabeçalhos da API de Xen do Linux

6.8. Man-pages-3.59

O pacote Man-pages contém mais de 1900 páginas de manual.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 23 MB

6.8.1. Instalação do Man-pages

Instale Man-pages rodando:

```
make install
```

6.8.2. Conteúdo do Man-pages

Arquivos instalados: várias páginas de manual

Breve descrição

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

6.9. Glibc-2.19

O pacote Glibc contém a biblioteca C principal. Este pacote prov# as rotinas básicas para alocação de memória, busca em diretórios, abrir e fechar arquivos, ler e escrever arquivos, manuseio de strings, correspondência de padrões, aritmética, e muito mais.

Tempo de Construção: 17.1 SBU
Espaço de disco: 922 MB

6.9.1. Instalação do Glibc

Nota

Alguns pacotes fora do LFS sugerem instalar a libiconv GNU para traduzir dados de uma codificação para outra. A página do projeto (*http://www.gnu.org/software/libiconv/*) diz "Esta biblioteca fornece uma implementação `iconv()`, para uso em sistemas que não tem um, ou cujas implementações não conseguem converter de/para Unicode." Glibc fornece uma implementação `iconv()` e pode converter de/para Unicode, portanto libiconv não é necessário em um sistema LFS.

Primeiro conserte um pequeno problema quando instalando o script tzselect:

```
sed -i 's/\\$$(pwd)/`pwd`/' timezone/Makefile
```

Alguns programas do Glibc usam diretórios `/var/db` que não estão em conformidade com o FHS para armazenar seus dados em tempo de execução. Aplique o patch a seguir para fazer tais programas armazenarem seus dados de tempo de execução em localizações que estejam em conformidade com o FHS:

```
patch -Np1 -i ../glibc-2.19-fhs-1.patch
```

O sistema de compilação do Glibc é independente e irá instalar perfeitamente, mesmo que os arquivos de especificação do copilador ainda estejam apontando para `/tools`. As especificações e o linker não podem ser ajustados antes da instalação do Glibc porque os testes do autoconf do Glibc dão resultados falsos e destroem o objetivo de ter uma compilação limpa.

A documentação do Glibc recomenda construir o Glibc fora do diretório onde está o código fonte (source) em um diretório dedicado a contrução (build):

```
mkdir -v ../glibc-build
cd ../glibc-build
```

Prepare o Glibc para compilação:

```
../glibc-2.19/configure    \
    --prefix=/usr          \
    --disable-profile      \
    --enable-kernel=2.6.32 \
    --enable-obsolete-rpc
```

O significado das novas opções do configure:

`--enable-obsolete-rpc`
Instala os cabeçalhos NIS e RPC que não são instalados por padrão; eles são necessários para reconstruir Glibc e por vários pacotes BLFS.

Compile o pacote:

```
make
```

Importante

Nesta seção, a suite de testes do Glibc é considerada crítica. Não pule sob qualquer circunstância.

Geralmente alguns testes não passam, mas você pode ignorar quaisquer das falhas abaixo. Agora teste os resultados da compilação:

```
make -k check 2>&1 | tee glibc-check-log
grep Error glibc-check-log
```

ocê provavelmente verá uma falha esperada (ignorada) nos testes *posix/annexc* e *conform/run-conformtest*. Em adição, a suite de testes do Glibc é um pouco dependente do sistema anfitrião. Esta é uma lista dos problemas mais comuns:

- Os testes *nptl/tst-clock2*, *nptl/tst-attr3*, *tst/tst-cputimer1*, e *rt/tst-cpuclock2* são conhecidos por falhar. A razão não é totalmente entendida, mas há indicações de que problemas de temporização podem desencadear essas falhas.
- Os testes com "math" às vezes falham quando são executados em sistemas onde a CPU não é um processador novo e genuíno Intel ou AMD.
- Quando rodar em máquinas mais velhas e lentas ou em sistemas sob carga, alguns testes podem falhar devido às limitações de tempo de teste serem excedidas. Modificar o comando "make check" para configurar um "TIMEOUTFACTOR" já foi reportado como sendo uma solução para eliminar esses erros (e.g. **TIMEOUTFACTOR=16 make -k check**).
- posix/tst-getaddrinfo4 sempre falhará devido a ausência de uma conexão de rede durante a execução dos testes.
- libio/tst-ftell-partial-wide.out falha porque precisa de um locale que ainda não foi gerado.
- Outros testes conhecidos por falhar em algumas arquiteturas são posix/bug-regex32, misc/tst-writev, elf/check-textrel, nptl/tst-getpid2, nptl/tst-robust8, e stdio-common/bug22.

Mesmo sendo uma mensagem inofensiva, a instalação do Glibc irá reclamar da falta do `/etc/ld.so.conf`. Previna este alerta:

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

Instale o pacote:

```
make install
```

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

```
cp -v ../glibc-2.19/nscd/nscd.conf /etc/nscd.conf
mkdir -pv /var/cache/nscd
```

Os "locales" que podem fazer o sistema responder em uma língua diferente não foram instalados pelo comando acima. Nenhum dos "locales" é necessário, mas se alguns deles estiverem faltando, a suite de testes de futuros pacotes pode pular testes importantes.

Pode-se fazer a instalação de "locales" individuais usando o programa **localedef**. E.g., o primeiro comando abaixo **localedef** combina a definição de locale independente de caracter `/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 instruções seguintes irão instalar o conjunto mínimo de locales necessário para uma cobertura ótima dos testes:

```
mkdir -pv /usr/lib/locale
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 en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
```

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

Alternativamente, instala todos os locales listados no arquivo `glibc-2.19/localedata/SUPPORTED` (inclui cada locale listado acima e muito mais) de uma só vez com o comando que consome muito tempo:

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

Use o comando **localedef** para criar e instalar locales não listados no arquivo `glibc-2.19/localedata/SUPPORTED` no caso pouco provável de você precisar.

6.9.2. Configurando o Glibc

O arquivo `/etc/nsswitch.conf` precisa ser criado porque, mesmo que o Glibc disponha de padrões quando este arquivo estiver faltando ou corrompido, esses padrões não funcionam bem em um ambiente em rede. O fuso horário também precisa ser configurado.

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
```

Instale informações do fuso horário:

```
tar -xf ../tzdata2013i.tar.gz

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

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

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

O significado do comando zic:

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

Isso cria fusos horários posix, sem qualquer segundo bissesto. É convensão colocá-los em ambos `zoneinfo` e `zoneinfo/posix`. É necessário colocar os fusos horários POSIX em `zoneinfo`, do contrário várias suites de testes reportarão erros. Em um sistema embarcado, onde o espaço é apertado e você não pretende atualizar o fuso horário, você pode salvar 1.9MB não usando o diretório `posix`, mas algumas aplicações ou suites de testes podem retornar resultados não tão bons

`zic -L leapseconds ...`

Isso cria os fusos horários corretos, incluindo segundos bissextos. Em um sistema embarcado, onde espaço é apertado e você não pretende atualizar o fuso horário, ou sequer se importam com a hora correta, você pode salvar 1.9 MB omitindo o diretório `right`.

`zic ... -p ...`
Isso cria o arquivo `posixrules`. Nós usamos "New Yourk" porque o POSIX requer que as regras de horário estejam de acordo com as regras americanas.

Uma maneira de determinar o fuso horário local é rodando o seguinte script:

```
tzselect
```

Depois de responder algumas perguntas sobre a localização, o script irá retornar o nome do zona do fuso horário (e.g., *America/Edmonton*). Há também outros fusos horários possíveis 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` rodando:

```
cp -v /usr/share/zoneinfo/<xxx> /etc/localtime
```

Substitua *<xxx>* com o nome do fuso horário escolhido (e.g., Canada/Eastern).

6.9.3. Configurando o Vinculador Dinâmico (linker dinâmico)

Por padrão, o vinculador dinâmico (`/lib/ld-linux.so.2`) procura em `/lib` e `/usr/lib` pelas bibliotecas dinâmicas que são necessárias para os programas que eles estão rodando. Entretanto, se houver bibliotecas em diretórios diferentes de `/lib` e `/usr/lib`, elas precisam ser adicionadas ao arquivo `/etc/ld.so.conf` para que o vinculador dinâmico possa encontrá-las. Dois diretórios que são conhecidos por conterem bibliotecas adicionais são `/usr/local/lib` e `/opt/lib`, então adicione esses diretórios ao caminho de busca do vinculador 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 desejar, o vinculador dinâmico também pode buscar um diretório e incluir o conteúdo de arquivos encontrado lá. Geralmente os arquivos nesse diretório include contém uma linha especificando o caminho para a biblioteca desejada. Para adicionar esta habilidade rode 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
```

6.9.4. Conteúdo do Glibc

Programas instalados: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, makedb, mtrace, nscd, pcprofiledump, pldd, rpcgen, sln, sotruss, sprof, tzselect, xtrace, zdump, e zic

Bibliotecas instaladas: ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libpthread_nonshared.a, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so, e libutil.{a,so}

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

Breve descrição

catchsegv	Pode ser usado para obter informações de pilha quando um programa termina com um erro de segmentação
gencat	Gera um catálogo de mensagens
getconf	Exibe os valores de configuração do sistema para variáveis específicas do sistema de arquivos
getent	Obtém entradas de uma base de dados administrativa
iconv	Executa conversão de conjuntos de caracteres
iconvconfig	Cria arquivos de configuração de módulos de carregamento rápido **iconv**
ldconfig	Configura as ligações de tempo de execução do vinculador dinâmico
ldd	Reporta que bibliotecas compartilhadas são necessárias para cada dado programa ou biblioteca compartilhada.
lddlibc4	Auxilia **ldd** com arquivos objeto
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 trace de memória e mostra um sumário em formato legível por um humano
nscd	É um daemon que provê um cache para os pedidos de serviços de nomes mais comuns
pcprofiledump	Exibe informação gerada por perfis PC
pldd	Lista os objetos dinâmicos compartilhados usados pelos processos em execução
rpcgen	Gera código C para implementar o protocolo "Remote Procedure Call" (RPC)
sln	Um programa **ln** estaticamente vinculado
sotruss	Identifica chamadas de procedimentos de bibliotecas compartilhadas de um comando específico

sprof	Lê e exibe informações de perfil de objetos compartilhados
tzselect	Pergunto ao usuário sobre a localização do sistema e reporta a descrição do fuso horário correspondente
xtrace	Investiga a execução de um programa exibindo a função atualmente executada
zdump	Exibe o fuso horário
zic	O compilador do fuso horário
`ld.so`	O programa auxiliar para executáveis de bibliotecas compartilhadas
`libBrokenLocale`	Usado internamente pelo Glibc como um hack para fazer com que programas quebrados funcionem (e.g., algumas aplicações Motif). Para mais informações, veja comentário em `glibc-2.19/locale/broken_cur_max.c`
`libSegFault`	O manipulador de sinais de falha de segmentação, usado por **catchsegv**
`libanl`	Biblioteca assíncrona de pesquisa de nomes
`libc`	A biblioteca C principal
`libcidn`	Usado internamente pelo Glibc para manusear nomes de domínios internacionalizados na função `getaddrinfo()`
`libcrypt`	A biblioteca de criptografia
`libdl`	A biblitoeca de interface do vinculador dinâmico
`libg`	Biblioteca de mentira que não contém funções. Anteriormente era uma biblioteca de tempo de execução para **g++**
`libieee`	Vincular este módulo força regras de tratamento de erros para funções matemáticas como definido pelo "Institute of Electrical and Electronic Engineers" (IEEE). O padrão é tratamento de erro POSIX.1
`libm`	A biblioteca matemática
`libmcheck`	Aciona checagem de alocação de memória quando vinculado a
`libmemusage`	Usado por **memusage** para ajudar a coletar informações sobre o uso de memória por um programa
`libnsl`	A biblioteca de serviços de rede
`libnss`	A biblioteca "Name Service Switch", contendo funções para resolver nomes de hosts, nomes de usuários, nomes de grupos, pseudônimos, serviços, protocolos, etc.
`libpcprofile`	Contém funções de perfil usadas para investigar a quantidade de tempo de CPU gasta em linhas de código específicas
`libpthread`	A biblioteca POSIX de "threads"
`libresolv`	Contém funções para criação, envio e interpretação de pacotes para os servidores de nomes de domínios da internet
`librpcsvc`	Contém funções que provêm vários serviços RPC
`librt`	Contém funções que provém muitas das interfaces especificadas pela Extensão de Tempo Real POSIX.1b
`libthread_db`	Contém funções para construir depuradores para programas multi-threaded
`libutil`	Contém código para funções “standard” usadas em muitos utilitários Unix

6.10. Ajustando a Toolchain

Agora que a biblioteca C final foi instalada, é hora de ajustar a toolchain de mo de que ela ligará qualquer novo programa compilado a essas novas bibliotecas.

Primeiro, faça o backup do linker em `/tools` e substitua o mesmo pelo linker ajustado que nós fizemos no capítulo 5. Nós também criaremos um link para sua duplicata em `/tools/$(gcc -dumpmachine)/bin`:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(gcc -dumpmachine)/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(gcc -dumpmachine)/bin/ld
```

A seguir, nós emendamos os arquivos de especificação do GCC de modo que eles apontem para o novo linker dinâmico. Simplesmente deletando todas as instâncias de "/tools" deve nos deixar com o caminho correto para o linker dinâmico. Também ajuste os arquivos de especificação de modo que o GCC saiba onde encontrar os cabeçalhos corretos e os arquivos de inicialização do Glibc. Um comando **sed** faz isso:

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

É uma boa idéia inspecionar visualmente os arquivos de especificação para verificar se as mudanças pretendidas foram realmente executadas.

É imperativo neste ponto assegurar-se que as funções básicas (compilar e linkar) da toolchain ajustada estejam funcionando como esperado. Para fazer isso, execute o seguinte teste:

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

Se tudo estiver funcionando corretamente, não deve haver qualquer erro, e a saída do último comando será (com diferenças dependendo da plataforma no nome do linker dinâmico):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

Note que `/lib` agora é o prefixo do nosso linker dinâmico.

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

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

Se tudo estiver funcionando corretamente, não deve haver quaisquer erros, e a saída do último comando será:

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

Verifique que o compilador está buscando pelo arquivos de cabeçalho corretos:

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

Esse comando deve resultar em sucesso com a seguinte saída:

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

A seguir, verifique que o novo linker está sendo usado com o path de busca correto:

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

Se tudo estiver funcionando corretamente, não deve haver quaisquer erros, e a saída do último comando será:

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

Agora certifique-se que nós estamos usando a libc correta:

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

Se tudo estiver funcionando corretamente, não deve haver quaisquer erros, e a saída do último comando (permitindo um diretório lib64 em hosts de 64-bit) será:

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

Por último, certifique-se que o GCC esteja usando o linker dinâmico correto:

```
grep found dummy.log
```

Se tudo estiver funcionando corretamente, não deve haver qualquer erro, e a saída do último comando será (com diferenças dependendo da plataforma no nome do linker dinâmico e um diretório lib64 em hosts de 64-bit):

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

Se a saída não aparece como mostrado acima ou não aparece de jeito nenhum, então há algo seriamente errado. Investigue e retrace os paços para encontrar onde o problema está e corrija o mesmo. A razão mais provável é que alguma coisa deu errado no ajuste dos arquivos de especificação. Qualquer problema precisa ser resolvido antes de continuar com o processo.

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

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

6.11. Zlib-1.2.8

O pacote Zlib contém rotinas para compressão e descompressão usadas por alguns programas.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 4.6 MB

6.11.1. Instalação do Zlib

Prepare o Zlib para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

A biblioteca compartilhada precisa der movida para `/lib`, e como resultado o arquivo `.so` em `/usr/lib` precisará ser recriado:

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

6.11.2. Conteúdo do Zlib

Bibliotecas instaladas: libz.{a,so}

Breve descrição

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

6.12. File-5.17

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

Tempo de Construção: 0.1 SBU
Espaço de disco: 12.5 MB

6.12.1. Instalação do File

Prepare o File para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.12.2. Conteúdo do File

Programas instalados: arquivo
Biblioteca instalada: libmagic.so

Breve descrição

file	Tenta classificar cada arquivo dado; ele faz isso executando vários testes—testes de sistema de arquivos, testes de números mágicos, testes de linguagem
`libmagic.so`	Contém rotinas para reconhecimento de números máginas, usado pelo programa **file**

6.13. Binutils-2.24

O pacote Binutils contém um linker, um assembler, e outras ferramentas para manusear arquivos objeto.

Tempo de Construção: 2.0 SBU
Espaço de disco: 365 MB

6.13.1. Instalação do Binutils

Verifique se os PTYs estão funcionando propriamente dentro do ambiente chroot executando um teste simples:

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

Este comando deve retornar o seguinte:

```
spawn ls
```

Se, ao invés, a saída incluir a mensagem abaixo, então o ambiente não está configurado para operação adequado de PTY. O problema precisa ser resolvido antes de executar a suite de testes do Binutils e GCC:

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

Suprime a instalação de um arquivo `standards.info` desatualizado uma vez que um novo é instalado posteriormente quando instalar Autoconf:

```
rm -fv etc/standards.info
sed -i.bak '/^INFO/s/standards.info //' etc/Makefile.in
```

A documentação do Binutils recomenda construir o Binutils fora do diretório onde está o código fonte (source) em um diretório dedicado a contrução (build)

```
mkdir -v ../binutils-build
cd ../binutils-build
```

Prepare Binutils para a compilação:

```
../binutils-2.24/configure --prefix=/usr --enable-shared
```

Compile o pacote:

```
make tooldir=/usr
```

O significado do parâmetro do make:

`tooldir=/usr`

Normalmente, o tooldir (o diretório onde os executáveis será localizado no final) é configurado para `$(exec_prefix)/$(target_alias)`. Por exemplo, máquinas x86_64 expandiriam isso para `/usr/x86_64-unknown-linux-gnu`. Devido ao foto deste ser um sistema customizado, este diretório de destino específico `/usr` não é necessário. `$(exec_prefix)/$(target_alias)` seria usado se o sistema usasse um compilador cruzado (por exemplo, compilar um pacote em uma máquina Intel que gera código que pode ser executado em máquinas PowerPC).

Importante

A suite de testes para Binutils nesta seção é considerada crítica. Não pule sob quaisquer circunstâncias.

Teste os resultados:

```
make check
```

Instale o pacote:

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

6.13.2. Conteúdo do Binutils

Programas instalados: addr2line, ar, as, c++filt, elfedit, gprof, ld, ld.bfd, nm, objcopy, objdump, ranlib, readelf, size, strings, e strip
Bibliotecas instaladas: libbfd.{a,so}, and libopcodes.{a,so}
Diretório instalado: /usr/lib/ldscripts

Breve descrição

addr2line Traduz endereços de programas para nomes de arquivos e número de linhas; dado um endereço e o nome de um executável, ele usa a informação de depuração no executável para determinar que arquivo fonte e número de linha estão associados ao endereço

ar Cria, modifica e extrai alguns arquivos

as Um assembler que monta a saída do **gcc** em arquivos objeto

c++filt Usado pelo linker para decodificar símbolos C++ e Java e para evitar conflitos de funções sobrecarregadas

elfedit Atualiza os cabeçalhos ELF dos arquivos ELF

gprof Exibe dados do perfil de gráficos de chamada

ld Um linker que combina um número de objetos e arquivos em um único arquivo, realocando seus dados e vinculando suas referências simbólicas

ld.bfd Um hard link para **ld**

nm Lista os símbolos que ocorrem em um dado arquivo-objeto.

objcopy Traduz um tipo de arquivo-objeto em outro

objdump Exibe informação sobre um dado arquivo-objeto, com opções controlando a informação 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 arquivo e armazena este índice no arquivo; o índice lista todos os símbolos definidos pelos membros do arquivo que são arquivos-objeto que podem ser realocados

readelf Exibe informações sobre binários do tipo ELF

size Lista o tamanho da seção e o tamanho total de um dado arquivo-objeto

strings Exibe, para cada arquivo, a seqüência imprimível de caracteres que são no mínimo do tamanho especificado (padronizado para quatro); para arquivos-objeto, ele imprime, por padrão, apenas as strings das seções de inicialização e carregamento enquanto que para outros tipos de arquivos, ele escaneia o arquivo completo

strip	Descarta símbolos de arquivos objeto
`libbfd`	A biblioteca de descrição de arquivos binários
`libopcodes`	Uma biblioteca para lidar com opcodes—a versão de “texto legível” de instruções para o processador; é usado para construir utilitários como **objdump**.

6.14. GMP-5.1.3

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

Tempo de Construção: 1.2 SBU
Espaço de disco: 50 MB

6.14.1. instalação do GMP

Nota

Se você estiver compilando para um sistema 32-bit x86, mas tem uma CPU capaz de rodar código 64-bit *e* você especificou `CFLAGS` no ambiente, o script configure tentará configurar para 64-bits falhará. Avoid this by invoking the configure command below with

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

Prepare o GMP para a compilação:

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

O significado das novas opções do configure:

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

Compile o pacote:

```
make
```

Importante

A suite de testes para GMP nesta seção é considerada crítica. Não pule sob quaisquer circunstâncias.

Teste os resultados:

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

Certifique-se que todos os 185 testes na suite de testes passaram. Verifique o resultado rodando o seguinte comando:

```
awk '/tests passed/{total+=$2} ; END{print total}' gmp-check-log
```

Instale o pacote:

```
make install
```

Se desejar, instale a documentação:

```
mkdir -v /usr/share/doc/gmp-5.1.3
cp    -v doc/{isa_abi_headache,configuration} doc/*.html \
         /usr/share/doc/gmp-5.1.3
```

6.14.2. Conteúdo do GMP

Bibliotecas Instaladas: libgmp.{a,so} and libgmpxx.{a,so}
Diretório instalado: /usr/share/doc/gmp-5.1.3

Breve descrição

`libgmp` Contém funções matemáticas de precisão.

`libgmpxx` Contém funções matemáticas de precisão em C++.

6.15. MPFR-3.1.2

O pacote MPFR contém funções para matemática de precisão múltipla.

Tempo de Construção: 0.8 SBU
Espaço de disco: 27 MB

6.15.1. Instalação do MPFR

Prepare o MPFR para a compilação:

```
./configure --prefix=/usr        \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-3.1.2
```

Compile o pacote:

```
make
```

Importante

A suite de testes para MPFR nesta seção é considerada crítica. Não pule sob quaisquer circunstâncias.

Teste os resultados e certifique-se que todos os testes passaram:

```
make check
```

Instale o pacote:

```
make install
```

Instale a documentação

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

6.15.2. Conteúdo do MPFR

Bibliotecas Instaladas: libmpfr.{a,so}
Diretório instalado: /usr/share/doc/mpfr-3.1.2

Breve descrição

`libmpfr` Contém funções matemáticas de precisão múltipla.

6.16. MPC-1.0.2

O pacote MPC contém uma biblioteca para aritmética de números complexos com alta precisão arbitrária e correto arredondamento de resultados.

Tempo de Construção: 0.4 SBU
Espaço de disco: 10.2 MB

6.16.1. Instalação do MPC

Prepare o MPC para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.16.2. Conteúdo do MPC

Bibliotecas Instaladas: libmpc.{a,so}

Breve descrição

`libmpc` Contém funções matemáticas complexas

6.17. GCC-4.8.2

O pacote GCC contém o Gnu Compiler Collection, que inclui os compiladores C e C++.

Tempo de Construção: 55.6 SBU
Espaço de disco: 2.2 GB

6.17.1. Instalação do GCC

Como em Seção 5.10, “GCC-4.8.2 - Pass 2”, aplique o seguinte **sed** para forçar a compilação a usar a flag `-fomit-frame-pointer` para garantir compilações consistentes:

```
case `uname -m` in
  i?86) sed -i 's/^T_CFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in ;;
esac
```

Além disso conserte um erro em uma das verificações dos Makefiles e desabilite um teste na sauite de testes do g++ libmudflap:

```
sed -i -e /autogen/d -e /check.sh/d fixincludes/Makefile.in
mv -v libmudflap/testsuite/libmudflap.c++/pass41-frag.cxx{,.disable}
```

A documentação do GCC recomenda construir o GCC fora do diretório source em um diretório build dedicado:

```
mkdir -v ../gcc-build
cd ../gcc-build
```

Prepare o GCC para a compilação:

```
SED=sed                              \
../gcc-4.8.2/configure               \
     --prefix=/usr                   \
     --enable-shared                 \
     --enable-threads=posix          \
     --enable-__cxa_atexit           \
     --enable-clocale=gnu            \
     --enable-languages=c,c++        \
     --disable-multilib              \
     --disable-bootstrap             \
     --with-system-zlib
```

Note que para outras línguas, há alguns pré-requisitos que não estão disponíveis. Veja o livro BLFS para instruções sobre como construir todas as línguas suportadas pelo GCC.

O significado da nova opção de configuração:

`SED=sed`
Definindo essa variável de ambiente evida uma codificação definitiva para o caminho /tools/bin/sed.

`--with-system-zlib`
Esse parâmetro diz ao GCC para linkar a cópia da biblioteca Zlib do sistema, em vez de sua própria cópia interna.

Compile o pacote:

```
make
```

Importante

Nesta seção, a suite de testes do GCC é considerada crítica. Não pule sob qualquer circunstância.

Um conjunto de testes na suite de testes do GCC é conhecida por esgotar a pilha, então aumente o espaço de pilha para rodar os testes:

```
ulimit -s 32768
```

Teste os resultados, mas não pare nos erros:

```
make -k check
```

Para receber um sumário dos resultados da suite de testes, rode:

```
../gcc-4.8.2/contrib/test_summary
```

Para ter apenas os sumários, faça a saída passar por **`grep -A7 Summ`**.

Os resultados podem ser comparados com aqueles encontrados em *http://www.linuxfromscratch.org/lfs/build-logs/7.5/* e *http://gcc.gnu.org/ml/gcc-testresults/*.

Umas poucas falhas inesperadas não podem ser evitadas sempre. Os desenvolvedores do GCC geralmente estão cientes dessas questões, mas ainda não as resolveram. Em particular, os testes da `libmudflap` são conhecidos por serem particularmente problemáticos como resultado de um bug no GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). A menos que os resultados dos testes sejam amplamente diferentes daqueles nas URL acima, é seguro continuar.

Instale o pacote:

```
make install
```

Alguns pacotes esperam que o preprocessador C esteja instalado no diretório `/lib`. Para dar suporte a esses pacotes, crie um symlink:

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

Muitos pacotes usam o nome **cc** para chamar o compilador C. Para satisfazer esses pacotes, crie o symlink:

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

Agora que nossa toolchain final está no lugar, é importante certificar-se novamente que compilação e linkedição irão funcionar como esperado. Nós fazemos isso executando o mesmo teste que nós fizemos anteriormente neste capítulo:

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

Se tudo estiver funcionando corretamente, não deve haver qualquer erro, e a saída do último comando será (com diferenças dependendo da plataforma no nome do linker dinâmico):

```
[Requesting program interpreter: /lib/ld-linux.so.2]
```

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

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

Se tudo estiver funcionando corretamente, não deve haver quaisquer erros, e a saída do último comando será:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.8.2/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.8.2/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.8.2/../../../crtn.o succeeded
```

Dependendo da arquitetura de sua máquina, a saída acima pode diferir levemente, a diferença geralmente é o nome do diretório depois de `/usr/lib/gcc`. Se sua máquinas é um sistema de 64-bit, você pode ver também um diretóriode de nome `lib64` no final da string. A coisa importante a se olhar aqui é se o **gcc** encontrou todos os três arquivos `crt*.o` sob o diretório `/usr/lib`.

Verifique que o compilador está buscando pelo arquivos de cabeçalho corretos:

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

Esse comando deve resultar em sucesso com a seguinte saída:

```
#include <...> search starts here:
 /usr/lib/gcc/i686-pc-linux-gnu/4.8.2/include
 /usr/local/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.8.2/include-fixed
 /usr/include
```

Novamente, note que o diretório nomeado de acordo com seu "target triplet" pode ser diferente do acima, dependendo de sua arquitetura.

Nota

Com na versão 4.3.0, GCC agora instala incondicionalmente o arquivo `limits.h` dentro do diretório privado `include-fixed` e esse diretório precisa estar no lugar.

A seguir, verifique que o novo linker está sendo usado com o path de busca correto:

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

Se tudo estiver funcionando corretamente, não deve haver quaisquer erros, e a saída do último comando será:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Um sistema 64-bir pode ser um pouco mais de diretórios. Por exemplo, aqui está a saída de uma máquina x86_64:

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

Agora certifique-se que nós estamos usando a libc correta:

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

Se tudo estiver funcionando corretamente, não deve haver quaisquer erros, e a saída do último comando (permitindo um diretório lib64 em hosts de 64-bit) será:

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

Por último, certifique-se que o GCC esteja usando o linker dinâmico correto:

```
grep found dummy.log
```

Se tudo estiver funcionando corretamente, não deve haver qualquer erro, e a saída do último comando será (com diferenças dependendo da plataforma no nome do linker dinâmico e um diretório lib64 em hosts de 64-bit):

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

Se a saída não aparece como mostrado acima ou não aparece de jeito nenhum, então há algo seriamente errado. Investigue e retrace os paços para encontrar onde o problema está e corrija o mesmo. A razão mais provável é que alguma coisa deu errado no ajuste dos arquivos de especificação. Qualquer problema precisa ser resolvido antes de continuar com o processo.

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

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

Finalmente, mova um arquivo colocado no lugar errado:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

6.17.2. Conteúdo do GCC

Programas instalados:	c++, cc (link to gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, and gcov
Bibliotecas instaladas:	libasan.{a,so}, libatomic.{a,so}, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libiberty.a, libitm.{a,so}, liblto_plugin.so, libmudflap.{a,so}, libmudflapth.{a,so}, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libsupc++.a e libtsan.{a,so}
Diretórios instalado:	/usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc, /usr/share/gcc-4.8.2

Breve descrição

c++	O compilador C++
cc	O compilador C
cpp	O preprocessador C; é usado pelo compilador para expandir as sentenças #include, #define e similares nos arquivos fonte
g++	O compilador C++
gcc	O compilador C
gcc-ar	Um encapsulador para **ar** que adiciona um plugin a linha de comando. Este programa é usado apenas para adicionar "link time optization" e não é útil com as opções de compilação padrão.

gcc-nm	Um encapsulador para **nm** que adiciona um plugin a linha de comando. Este programa é usado apenas para adicionar "link time optization" e não é útil com as opções de compilação padrão.
gcc-ranlib	Um encapsulador para **ranlib** que adiciona um plugin a linha de comando. Este programa é usado apenas para adicionar "link time optization" e não é útil com as opções de compilação padrão.
gcov	Uma ferramenta de teste de otimização; usada para analisar programas para determinar onde as otimizações terão maior efeito
libasan	A Biblioteca de Sanitização de Endereço em tempo de execução
`libgcc`	Contém suporte em tempo de execução para o **gcc**
`libgcov`	A biblioteca é linkada a um programa quando quando o GCC é instruído a habilitar "profiling"
`libgomp`	Implementação GNU da API OpenMP para programação em paralelo de memória compartilhada e multiplataforma em C/C++ e Fortran
`libiberty`	Contains routines used by various GNU programs, including **getopt**, **obstack**, **strerror**, **strtol**, e **strtoul**
`liblto_plugin`	GCC plugin Link Time Optimization (LTO) que permite ao GCC executar otimizações através de unidades de compilação.
`libmudflap`	Contém rotinas que suportam funcionalidade de checagem de limites do GCC
`libquadmath`	GCC API "Quad Precision Math Library" (Biblioteca de Precisão Quádrupla)
`libssp`	Contém rotinas que dão suporte a funcionalidade de proteção contra "stack-smashing" do GCC
`libstdc++`	A biblioteca C++ padrão
`libsupc++`	provê rotinas de suporte a linguagem de programação C++
`libtsan`	A Biblioteca de Sanitização de Thread em tempo de execução

6.18. Sed-4.2.2

O pacote Sed contém um editor de fluxo.

Tempo de Construção: 0.2 SBU
Espaço de disco: 6.7 MB

6.18.1. Instalação do Sed

Prepare o Sed para a compilação:

```
./configure --prefix=/usr --bindir=/bin --htmldir=/usr/share/doc/sed-4.2.2
```

O significado da nova opção de configuração:

`--htmldir`
Isso configura o diretório onde a documentação em HTML será instalada.

Compile o pacote:

```
make
```

Gera a documentação em HTML

```
make html
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Instale a documentação em HTML:

```
make -C doc install-html
```

6.18.2. Conteúdo do Sed

Programa instalado: sed
Diretório instalado: /usr/share/doc/sed-4.2.2

Breve descrição

sed Filtra e transforma arquivos de texto em uma única passagem

6.19. Bzip2-1.0.6

O pacote Bzip2 contém programas para compressão e descompressão de arquivos. Comprimir arquivos de texto com **bzip2** gera melhores taxas de compressão do que com o tradicional **gzip**.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 6.9 MB

6.19.1. Instalação do Bzip2

Aplique um patch que irá instalar a documentação para este pacote:

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch
```

O comando seguinte garante a instalação dos links simbólicos seja relativa:

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

Garante que as páginas de manual sejam instaladas na localização correta:

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

Prepare o Bzip2 para compilação com:

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

O significado do parâmetro do make:

`-f Makefile-libbz2_so`

Isso fará com que o Bzip2 compile usando um arquivo `Makefile` diferente, neste caso o arquivo `Makefile-libbz2_so` que cria a biblioteca dinâmica `libbz2.so` e linka os utilitários do Bzip2 a ela.

Compile e teste o pacote:

```
make
```

Instale os programas:

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

Instale o binário compartilhado **bzip2** no diretório `/bin`, certifique-se que alguns symlinks necessários sejam criados e faça uma limpeza:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.19.2. Conteúdo do Bzip2

Programas 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.{a,so}
Diretório instalado: /usr/share/doc/bzip2-1.0.6

Breve descrição

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 com seleção de blocos Burrows-Wheeler com codificação Huffman; a taxa de compressão é melhor do que aquela obtida usando compressores convencionais com o algoritmo “Lempel-Ziv” , como o **gzip**
bzip2recover	Tenta recuperar dados 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 implementa compressão com seleção de blocos sem perda usando o algoritmo Burrows-Wheeler

6.20. Pkg-config-0.28

O pacote pkg-config contém uma ferramenta para passar o path do include e/ou path de bibliotecas para ferramentas de compilação durante a execução do configure e do make.

Tempo de Construção: 0.4 SBU
Espaço de disco: 31 MB

6.20.1. Instalação do Pkg-config

Prepare o Pkg-config para a compilação:

```
./configure --prefix=/usr         \
            --with-internal-glib  \
            --disable-host-tool   \
            --docdir=/usr/share/doc/pkg-config-0.28
```

O significado das novas opções do configure:

`--with-internal-glib`
Isso permitirá que o pkg-config use sua versão interna do glib porque uma versão interna não está disponível no LFS.

`--disable-host-tool`
Esta opção desabilita a criação de um hard link indesejado para o programa pkg-config.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.20.2. Conteúdo do Pkg-config

Programa instalado: pkg-config
Diretório instalado: /usr/share/doc/pkg-config-0.28

Breve descrição

pkg-config retorna meta informação para a biblioteca ou pacote especificado.

6.21. Ncurses-5.9

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

Tempo de Construção: 0.6 SBU
Espaço de disco: 40 MB

6.21.1. Instalação do Ncurses

Prepare o Ncurses para a compilação:

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

O significado da opção do configure:

`--enable-widec`
Este parâmetro faz com que as bibliotecas 'wide-character' (e.g., `libncursesw.so.5.9`) sejam compiladas em vez das normais (e.g., `libncurses.so.5.9`). Essas bibliotecas 'wide-character' são úteis tanto em locales multibyte quando em tradicionais, enquanto as bibliotecas normais só funcionam em locales de 8-bits. Bibliotecas normais e wide-character não compatíveis em código, mas não são compatíveis em binários.

`--enable-pc-files`
Este switch gera e instala arquivos .pc para pkg-config.

Compile o pacote:

```
make
```

Este pacote contém uma suite de testes, entretanto ela só pode ser executada quando o pacote estiver instalado. O teste reside no diretório `test/`. Veja o arquivo `README` naquele diretório para maiores detalhes.

Instale o pacote:

```
make install
```

Mova as bibliotecas compartilhadas para o diretório `/lib`, onde espera-se que elas residam:

```
mv -v /usr/lib/libncursesw.so.5* /lib
```

Devido ao fato das bibliotecas terem sido movidas, um lsymlink aponta para um arquivo inexistente. Recrie-o:

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

Muitas aplicações ainda esperam que o linker seja capaz de encontrar bibliotecas Ncurses 'non-wide-character'. Ajuste tais aplicações para vincularem com bibliotecas 'wide-character' através de symlinks e scripts:

```
for lib in ncurses form panel menu ; do
    rm -vf                    /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv lib${lib}w.a      /usr/lib/lib${lib}.a
    ln -sfv ${lib}w.pc        /usr/lib/pkgconfig/${lib}.pc
done

ln -sfv libncurses++w.a /usr/lib/libncurses++.a
```

Finalmente, certifique-se de que aplicações antigas que procuram por `-lcurses` em tempo de compilação ainda sejam compiláveis:

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

Se desejar, instale a documentação do Ncurses:

```
mkdir -v       /usr/share/doc/ncurses-5.9
cp -v -R doc/* /usr/share/doc/ncurses-5.9
```

Nota

As instruções acima não criam bibliotecas 'non-wide-character' uma vez que nenhum pacote instalado através de compilação do código fonte vincularia a elas em tempo de execução. Se você tem que ter tais bibliotecas devido a algumas aplicações na forma de binário ou para estar em conformidade com LSB, compile o pacote novamente com os seguintes comandos:

```
make distclean
./configure --prefix=/usr    \
            --with-shared    \
            --without-normal \
            --without-debug  \
            --without-cxx-binding
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.21.2. Conteúdos do Ncurses

Programas instalados: captoinfo (link to tic), clear, infocmp, infotocap (link to tic), ncursesw5-config, reset (link to tset), tabs, tic, toe, tput, e tset

Bibliotecas instaladas: libcursesw.{a,so} (symlink and linker script to libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} and their non-wide-character counterparts without "w" in the library names.

Diretórios instalado: /usr/share/tabset, /usr/share/terminfo, /usr/share/doc/ncurses-5.9

Breve descrição

captoinfo	Converte uma descrição do termcap em uma descrição do terminfo
clear	Limpa a tela, se possível
infocmp	Compara ou imprime descrições terminfo
infotocap	Converte uma descrição terminfo em uma descrição termcap
ncursesw5-config	Dispõe informação de configuração para ncurses
reset	Reinicializa um terminal para seus valores padrão
tabs	Limpa e configura tabulações em um terminal
tic	O compilador de descrição de entrada terminfo que traduz um arquivo terminfo do formato fonte em formato binário necessário para as rotinas das bibliotecas ncurses. Um arquivo terminfo contém informações sobre as capacidades de um certo terminal
toe	Lista todos os tipos de terminal disponíveis, dando o nome primário e descrição para cada
tput	Faz os valores de capacidades dependentes do terminal disponíveis para o shell; também pode ser usado para resetar ou inicializar um terminal ou reportar seu nome longo
tset	Pode ser usado para inicializar terminais
`libcurses`	Um link para `libncurses`
`libncurses`	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
`libform`	Contém funções para implementar formulários
`libmenu`	Contém funções para implementar menus
`libpanel`	Contém funções para implementar painéis

6.22. Shadow-4.1.5.1

O pacote Shadow contém programas para manipulação de senhas de uma maneira segura.

Tempo de Construção: 0.2 SBU
Espaço de disco: 42 MB

6.22.1. Instalação do Shadow

Nota

Se você quiser reforçar o uso de senhas fortes, recorra a *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* para instalar CrackLib antes de compilar o Shadow. Então adicione *`--with-libcrack`* ao comando **configure** abaixo.

Desabilita a instalação do programa **groups** pe suas páginas de manual, uma vez que o Coreutils dispõe uma versão melhor:

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

Em vez de usar o método padrão *crypt*, use o método de encriptação de senha mais seguro *SHA-512* o qual permite senhas com mais de 8 caracteres. É também necessário mudar a localização obsoleta `/var/spool/mail` para caixas de correios de usuário que o Shadow usa por padrão pela localização `/var/mail` usada atualmente:

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

Nota

Se você escolher compilar o Shadow com suporte a CrackLib, rode o seguinte:

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

Prepare o Shadow para a compilação:

```
./configure --sysconfdir=/etc
```

Compile o pacote:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

Mova um programa mal colocado para sua localização adequada:

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

6.22.2. Configurando o Shadow

Este pacote contém utilitários para adicionar, modificar, e suprimir usuários e grupos; define e modifica suas senhas; e executa outras tarefas administrativas. Para uma explanação detalhada do que *password shadowing* significa, veja o arquivo `doc/HOWTO` dentro da árvore descompactada dos fontes. Se for usar o suporte ao Shadow, tenha na mente que os programas que necessitam de verificação de senhas (display managers, programas de ftp, pop3, etc..) devem ser compatíveis com o shadow. Isto é, eles precisam ser capazes de trabalhar com senhas Shadow.

Para permitir a proteção de senhas, execute o seguinte comando:

```
pwconv
```

Permitir a proteção de senhas de grupo, execute:

```
grpconv
```

A configuração padrão do Shadow para **useradd** tem algumas ressalvas que precisam de explicação. Primeiro, a ação padrão para **useradd** é criar o usuário e um grupo de mesmo nome. Por padrão o ID de usuário (UID) e o ID de grupo (GID) começarão por 1000. Isso significa que se você não passar parâmetros para **useradd**, cada usuário será membro de um único grupo no sistema. Se este comportamento não é desejado, você precisará passar o parâmetro *`-g`* para **useradd**. Os parâmetros padrão são armazenados no arquivo `/etc/default/useradd`. Você pode precisar modificar dois parâmetros neste arquivo para adequar a suas necessidades.

`/etc/default/useradd` Explicações de Parâmetros

`GROUP=1000`

Este parâmetro configura o início da numeração de grupos usado no arquivo /etc/group. Você pode modificá-lo para qualquer valor que deseje. Note que **useradd** nunca reutilizará um UID ou GID. Se o número identificado neste parâmetro está em uso, ele usará o próximo disponível após este. Note também que se você não tem um grupo 1000 no seu sistema a primeira vez que você usar **useradd** sem o parâmetro *`-g`*, você receberá uma mensagem no terminal que diz: `useradd: unknown GID 1000`. Você pode desconsiderar esta mensagem e o número de grupo 1000 será usado.

`CREATE_MAIL_SPOOL=yes`

Este parâmetro faz com que **useradd** crie um arquivo mailbox para o novo usuário. **useradd** tornará a posse desse arquivo para o grupo `mail` com permissões 0660. Se você preferir que esses arquivos de mailbox não sejam criados por **useradd**, execute o seguinte comando:

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

6.22.3. Configurando a senha do root

Escolha uma senha para o usuário *root* e defina com:

```
passwd root
```

6.22.4. Conteúdo do Shadow

Programas instalados:	chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), e vipw
Diretório instalado:	/etc/default

Breve descrição

chage Usado para alterar o número máximo de dias entre as mudanças obrigatórias de senha

chfn Usado para alterar o nome completo do usuário e outras informações

chgpasswd Usado para atualizar as senhas de grupo em lote

chpasswd Usado para atualizar as senhas de usuários em lote

chsh Usado para alterar o shell de login padrão do usuário

expiry Verifica e reforça a política de expiração de senha

faillog Usado para examinar o registro de falhas no login, definir um número máximo de falhas antes que uma conta de usuário seja bloqueado, ou zerar a contagem de falhas

gpasswd Usado para adicionar e excluir membros e administradores aos grupos

groupadd Cria grupos com os nomes dados

groupdel Deleta o grupo com os nomes dados

groupmems Permite a um usuário gerenciar sua filiação a grupos sem a necessidade de privilégios de superusuário.

groupmod Usado para alterar o nome ou o GID do grupo especificado

grpck Verifica a integridade de arquivos de grupo `/etc/group` e `/etc/gshadow`

grpconv Cria ou atualiza os arquivos de grupo do shadow a partir do arquivo de grupo normal

grpunconv Atualiza `/etc/group` a partir de `/etc/gshadow` e então o deleta

lastlog Reporta o login mais recente de todos os usuários ou de um determinado usuário

login É usado para estabelecer uma nova sessão com o sistema

logoutd É um daemon usado para reforçar restrições no tempo de login e portas

newgrp É usado para modificar o GID atual durante uma sessão de login

newusers É usado para criar ou atualizar um série inteira de contas de usuários

nologin Exibe uma mensagem que diz 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 um usuário ou grupo de contas

pwck Verifica a integridade dos arquivos de senha `/etc/passwd` e `/etc/shadow`

pwconv Cria ou atualiza o arquivo de senha shadow a partir de um arquivo de senha normal

pwunconv Atualiza `/etc/passwd` a partir de `/etc/shadow` e então o deleta

sg Executa um dado comando enquanto a GID do usuário está configurada para o grupo dado

su Roda um shell com IDs de usuário e grupo substituídos

useradd Cria um novo usuário com o nome dado, ou atualiza a informação padrão de novo usuário

userdel Deleta a conta de usuário dada

usermod Usado para modificar o nome de login do usuário informado na linha de comando, seu número de identificação de usuário (UID), o shell padrão, o grupo inicial, o diretório home, etc.

vigr Edita os arquivos `/etc/group` ou `/etc/gshadow`

vipw Edita os arquivos `/etc/passwd` ou `/etc/shadow`

6.23. Psmisc-22.20

O pacote Psmisc contém programas para mostrar informações sobre processos que estão rodando.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 4.2 MB

6.23.1. Instalação do Psmisc

Prepare o Psmisc para a compilação:

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

Compile o pacote:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

Finalmente, mova os programas **killall** e **fuser** para a localização especificada pela FHS:

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

6.23.2. Conteúdo do Psmisc

Programas instalados: fuser, killall, peekfd, prtstat, pstree, e pstree.x11 (link para pstree)

Breve descrição

fuser	Mostra os IDs dos processos (PIDs) que usam dados arquivos ou arquivos do sistema
killall	Mata processos pelo nome; emite um sinal a todos os processos rodando quaisquer dos comandos dados
peekfd	Verifica arquivos descritores de um processo em execução, dado seu PID
prtstat	Imprime informação sobre um processo
pstree	Exibe processos em execução em formato de árvore
pstree.x11	O mesmo que **pstree**, exceto que ele espera por uma confirmação antes de sair

6.24. Procps-ng-3.3.9

O pacote Procps-ng contém programas para monitorar processos.

Tempo de Construção: 0.2 SBU
Espaço de disco: 13 MB

6.24.1. Instalação do Procps-ng

Agora prepare procps-ng para a compilação:

```
./configure --prefix=/usr                            \
            --exec-prefix=                           \
            --libdir=/usr/lib                        \
            --docdir=/usr/share/doc/procps-ng-3.3.9 \
            --disable-static                         \
            --disable-kill
```

O significado das opções do configure:

`--disable-kill`
Este switch desabilita a compilação do comando kill que foi instalado no pacote util-linux.

Compile o pacote:

```
make
```

A suite de testes precisa de algumas customizações para o LFS. Remova o teste que falha quando o script não usa um dispositivo tty. Para rodar a suite de testes, execute os comandos a seguir:

```
sed -i -r 's|(pmap_initname)\\\$|\1|' testsuite/pmap.test/pmap.exp
make check
```

Instale o pacote:

```
make install
```

Finalmente mova arquivos essenciais para uma localização que pode ser encontrada se `/usr` não estiver montado.

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

6.24.2. Conteúdo do Procps-ng

Programas instalados: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w, e, watch
Biblioteca instalada: libprocps.so

Breve descrição

free Reporta a quantidade total de memória livre e usada (tanto física quando swap) no sistema

pgrep Procura por processos baseado em seus nomes e outros atributos

pidof Procura por processos baseado em seus nomes e outros atributos

pkill Sinaliza processos baseado no nome e outros atributos

pmap Reporta o mapeamento da memória de um dado processo

ps Lista os processos atualmente em execução

pwdx Reporta o diretório de trabalho de um processo

slabtop Exibe informações detalhadas e em tempo real sobre cache do kernel

sysctl Modifica parâmetros do kernel em tempo de execução

tload Imprime um gráfico da média de carga atual do sistema

top Mostra uma lista dos processos mais intensos da CPU; isto fornece uma estimativa da atividade do processador central em tempo real

uptime Relata há quanto tempo o sistema está ligado, quantos usuários estão logados, e a média de carga no sistema

vmstat Reporta estatísticas sobre memória virtual, fornecendo informações sobre processos, memória, paginação, Input/Output (IO) de blocos, traps, e atividade da CPU

w Mostra que usuários estão atualmente logados, onde e desde quando

watch Roda um dado comando repetidamente, exibindo a primeira saída; isso permite que um usuário veja a mudança da saída com o decorrer do tempo

`libprocps` Contém as funções usadas pela maioria dos programas neste pacote

6.25. E2fsprogs-1.42.9

O pacote E2fsprogs contém utilitários para manipular o sistema de arquivos `ext2`. Ele também suporta os sistemas com journaling `ext3` e `ext4`.

Tempo de Construção: 1.7 SBU
Espaço de disco: 64 MB

6.25.1. Instalação do E2fsprogs

Primeiro conserte um problema com execução de testes de regressão no ambiente chroot LFS.

```
sed -i -e 's|^LD_LIBRARY_PATH.*|&:/tools/lib|' tests/test_config
```

A documentação do E2fsprogs recomenda que o pacote seja compilado em um subdiretório fora da árvore de diretórios do código fonte.

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

Prepare E2fsprogs para a compilação:

```
LIBS=-L/tools/lib                    \
CFLAGS=-I/tools/include              \
PKG_CONFIG_PATH=/tools/lib/pkgconfig \
../configure --prefix=/usr           \
             --with-root-prefix=""   \
             --enable-elf-shlibs     \
             --disable-libblkid      \
             --disable-libuuid       \
             --disable-uuidd         \
             --disable-fsck
```

O significado da variável de ambiente e das opções do configure:

`PKG_CONFIG_PATH, LIBS, CFLAGS`
Essas variáveis habilitam o e2fsprogs a compilar usando o pacote Seção 5.33, “Util-linux-2.24.1” construído anteriormente.

`--with-root-prefix=""`
Certos programas (como o programa **e2fsck**) são considerados programas essenciais. Quando, por exemplo, `/usr` não está montado, esses programas ainda precisam estar disponíveis. They belong in directories like `/lib` and `/sbin`. Se esta opção não for passada ao configure do E2fsprogs, os programas serão instalados no diretório `/usr`.

`--enable-elf-shlibs`
Isso cria as bibliotecas compartilhadas que alguns programas neste pacote usam.

`--disable-*`
Isso evita que E2fsprogs compile e instale as bibliotecas `libuuid` e `libblkid`, o daemon `uuidd`, e o encapsulador **fsck**, uma vez que o Util-Linux instalou todos eles previamente.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Um dos testes do E2fsprogs tentará alocar 256 MB dememória. Se você não tem significativamente mais RAM do que isso, certifique-se de habilitar espaço de memória swap para o teste. Veja Seção 2.3, “Criando um Sistema de Arquivos na Partição” e Seção 2.4, “Montando a Nova Partição” par detalhes sobre criar e habilitar espaço swap. Adicionalmente, três testes tentam alocar uma partição de dois terabytes e irão falhar a menos que você tenha pelo menos essa quantidade de espaço de disco disponível.

Instale os binários, documentação e bibliotecas compartilhadas:

```
make install
```

Instale as bibliotecas estáticas e os cabeçalhos:

```
make install-libs
```

Altere a permissão de escrita das bibliotecas estáticas instaladas de moto que os símbolos de depuração possam ser removidos posteriormente:

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Este pacote instala um arquivo `.info` mas não atualiza o arquivo `dir`. Descompacte este arquivo e atualize o aquivo `dir` do sistema usando os comandos a seguir.

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir /usr/share/info/libext2fs.info
```

Se desejar, crie e instale alguma documentação adicional executando os comandos a seguir:

```
makeinfo -o      doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir /usr/share/info/com_err.info
```

6.25.2. Conteúdo do E2fsprogs

Programas instalados:	badblocks, chattr, compile_et, debugfs, dumpe2fs,e2freefrag, e2fsck, e2image, e2label, e2undo, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, fsck.ext4dev, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mkfs.ext4dev, resize2fs, e tune2fs
Bibliotecas instaladas:	libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libquota.a, e libss.{a,so}
Diretório instalado:	/usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/quota, /usr/include/ss, /usr/share/et, /usr/share/ss

Breve descrição

badblocks	Busca por bad blocks em um dispositivo (geralmente uma partição de disco)
chattr	Muda os atributos de arquivos em um sistema de arquivos `ext2`; também muda sistemas de arquivos `ext3`, a versão com journaling do sistema de arquivos `ext2`
compile_et	Um compilador de tabela de erros; ele converte uma tabela de nomes de erros e códigos 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 informações sobe os superblocos e grupos de blocos para o sistema de arquivos presente em um dado dispositivo

e2freefrag Reporta informações sobre fragmentação de espaço livre

e2fsck É usado para verificar e, opcionalmente, reparar sistemas de arquivos `ext2` e `ext3`

e2image É usado para salvar informações críticas de um sistema de arquivos `ext2` em um arquivo

e2label Exibe ou muda o "label" no sistema de arquivos `ext2` presente em um dado dispositivo

e2undo Reexecuta o registro "desfazer" do undo_log para um sistema de arquivos ext2/ext3/ext4 encontrado em um dispositivo. Pode ser usado para desfazer uma operação que falhou em um programa do e2fsprogs.

e4defrag Desfragmentador online para sistemas de arquivo ext4

filefrag Reporta quão fragmentado um arquivo em particular pode estar

fsck.ext2 Por padrão verifica sistemas de arquivo `ext2`. Um hard link para **e2fsck**.

fsck.ext3 Por padrão verifica sistemas de arquivo `ext3`. Um hard link para **e2fsck**.

fsck.ext4 Por padrão verifica sistemas de arquivo `ext4`. Um hard link para **e2fsck**.

fsck.ext4dev Por padrão verifica sistemas de arquivo em desenvolvimento `ext4`. Um hard link para **e2fsck**.

logsave Salva a saída de um comando em um arquivo log

lsattr Lista os atributos de arquivos em um sistema de arquivos ext2

mk_cmds Converte uma tabela de nome de comandos e mensagens de ajuda em arquivo de código C adequado para uso com a biblioteca de subsistema `libss`

mke2fs Cria um sistema de arquivo `ext2` ou `ext3` em um dado dispositivo

mkfs.ext2 Por padrão cria um sistema de arquivos `ext2`. Um hard link para **mke2fs**.

mkfs.ext3 Por padrão cria um sistema de arquivos `ext3`. Um hard link para **mke2fs**.

mkfs.ext4 Por padrão cria um sistema de arquivos `ext4`. Um hard link para **mke2fs**.

mkfs.ext4dev Por padrão cria sistemas de arquivo em desenvolvimento `ext4`. Um hard link para **mke2fs**.

mklost+found Usado para criar um diretório `lost+found` em um sistema de arquivos `ext2`; ele pré-aloca blocos do disco para este diretório para deixar mais leve a tarefa do **e2fsck**

resize2fs Pode ser usado para aumentar ou reduzir um sistema de arquivos `ext2`

tune2fs Ajusta parâmetros do 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 programas a nível de usuário para manipular sistemas de arquivo `ext2`

`libquota` Fornece uma interface para criar e atualizar arquivos de quota e campos de superblocos ext4

`libss` Usado por **debugfs**

6.26. Coreutils-8.22

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

Tempo de Construção: 3.4 SBU
Espaço de disco: 116 MB

6.26.1. Instalação do Coreutils

POSIX requer que programas do Coreutils reconheçam limites de caracteres mesmo em locales multibyte. O patch seguinte conserta essa não-conformidade e outros bugs relacionados a internacionalização:

```
patch -Np1 -i ../coreutils-8.22-i18n-4.patch
```

Nota

No passado, muitos bugs foram encontrados neste patch. Quando reportar um novo bug para os mantenedores do Coreutils, por favor verifique primeiro se ele pode ser reproduzido sem este patch.

Agora prepare Coreutils para a compilação:

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \
            --prefix=/usr            \
            --enable-no-install-program=kill,uptime
```

O significado das opções do configure:

`--enable-no-install-program=kill,uptime`
O propósito deste switch é prevenir que o Coreutils instale que serão instalados por outros pacotes posteriormente.

Compile o pacote:

```
make
```

Pule para “Instalar o pacote” se voc# não vai rodar a suite de testes.

Agora a suite de testes está pronta para ser executada. Primeiro rode os testes que devem ser executados como usuário `root`:

```
make NON_ROOT_USERNAME=nobody check-root
```

Nós vamos rodar o resto dos testes como usuário `nobody`. Certos testes, entretanto, requerem que o usuário seja membro de mais de um grupo. Como estes testes não serão pulados nós iremos adicionar um grupo temporário e faremos o usuário `nobody` parte dele:

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Conserte algumas das premissões de modo que um usuário não-root possa compilar e rodar os testes:

```
chown -Rv nobody .
```

Agora rode os testes. Certifique-se que o PATH no ambiente **`su`** inclui /tools/bin.

```
su nobody -s /bin/bash \
          -c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

Remova o grupo temporário:

```
sed -i '/dummy/d' /etc/group
```

Instale o pacote:

```
make install
```

Mova os programas para as localizações especificadas pelo FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname,test,[} /bin
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i s/\"1\"/\"8\"/1 /usr/share/man/man8/chroot.8
```

Alguns dos scripts no pacote LFS-Bootscripts dependem de **head**, **sleep**, e **nice**. Como `/usr` pode não estar disponível durante os primeiros estágios da inicialização, esses binários precisam estar na partição raiz:

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.26.2. Conteúdo do Coreutils

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

Breve descrição

base64 Codifica e decodifica dados de acordo com a especificação base64 (RFC 3548)

basename Remove qualquer caminho e um dado sufixo do nome de um arquivo

cat Concatena arquivos para saída padrão

chcon Muda contexto de segurança para arquivos e diretórios

chgrp Muda o grupo a que pertencem arquivos e diretórios

chmod Muda as permissões de cada arquivo para a modalidade especificada; a modalidade pode ser uma representação simbólica das mudanças a fazer ou um número octal que representa as novas permissões

chown Muda a propriedade do usuário e/ou do grupo de arquivos e dos diretórios

chroot Roda um comando usando o diretório especificado como diretório `/`

cksum Exibe a assinatura CRC e contagem de bytes de cada arquivo especificado

comm Compara dois arquivos ordenados, exibindo tr#s colunas com 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 parâmetros dados ou números de linhas e exibindo a contagem de byte de cada novo arquivo

cut Exibe seções de linhas, selecionando as partes de acordo com dados campos ou posições

date Exibe data atual em um dado formato, ou configura a data do sistema

dd Copia um arquivo usando o tamanho de bloco e contagem, enquanto opcionalmente executa conversões nele

df Reporta a quantidade de espaço de disco disponível (e usada) em todos os sistemas de arquivos montados, ou apenas nos sistemas de arquivos que contém os arquivos selecionados

dir Lista o conteúdo de um dado diretório (o mesmo que o comando **ls**)

dircolors Emite comandos para configurar a variável de ambiente `LS_COLOR` epara mudar o esquema de cores usado por **ls**

dirname Remove o sufixo que não é diretório do nome de um arquivo

du Reporta a quantidade de espaço de disco usado pelo diretório atual, por cada diretório dado (incluindo subdiretórios) ou por cada um dos arquivos dados

echo Exibe as strings fornecidas

env Roda um comando em um ambiente modificado

expand Converte tabs 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 status indicando falha

fmt Reformata os parágrafos nos arquivos fornecidos

fold Quebra as linhas nos arquivos fornecidos

groups Reporta que grupos um usuário faz parte

head Prints the first ten lines (or the given number of lines) of each given file

hostid Reporta o número identificador (em hexadecimal) do host

id Exibe a ID de usuário, ID de grupo e os grupos dos quais o usuário atual, ou um outro usuário especificado, faz parde

install Copia arquivos enquanto configurando suas permissões e, se possível, seu dono e grupo

join Junta as linhas de dois arquivos separados que têm campos de junção idênticos

link Cria um hard link com o nome de um arquivo

ln Faz hard links ou um soft (symbolic) links entre arquivos

logname Exibe o nome de login do usuário atual

ls Lista o conteúdo de cada diretório dado

md5sum Exibe ou checa assinaturas Message Digest 5 (MD5)

mkdir Cria diretórios com os nomes dados

mkfifo Cria First-In, First-Outs (FIFOs), um “named pipe” no jargão UNIX, com os nomes dados

mknod Cria "device nodes" com os nomes fornecidos; um "device note" é um arquivo de caracteres especiais, um arquivo de blocos especiais, ou um FIFO

mktemp Cria arquivos temporários de uma forma segura; é usado em scripts

mv Move ou renomeia arquivos ou diretórios

nice Executa programas com prioridade modificada

nl Numera as linhas de um dado arquivo

nohup Roda um comando imune a interrupções, com sua saída redirecionada a um arquivo de log

nproc Imprime o número de unidades de processamento disponíveis em um processador

numfmt Converte números de ou para strings legíveis por humanos

od Despeja arquivos em octal ou em outros formatos

paste Une arquivos, juntado seqüencialmente linhas correspondentes lado a lado, separadas por caracteres tab

pathchk Verifica se os nomes de arquivos são válidos ou portáveis

pinky É um cliente finger leve; ele retorna algumas informações sobre um usuário fornecido

pr Pagina e organiza colunas de arquivos para impressão

printenv Exibe as variáveis de ambiente

printf Exibe os argumentos dados de acordo com uma formatação dada, muito parecido com a função printf do C

ptx Produz um índice permutado do conteúdo dos arquivos especificados, com cada palavra-chave em seu contexto

pwd Retorna o nome do diretório de trabalho atual

readlink Retorna o valor de um dado link simbólico

realpath Imprime o path resolvido

rm Remove arquivos ou diretórios

rmdir Remove diretórios se eles estiverem vazios

runcon Roda um comando com um contexto de segurança específico

seq Imprime uma seqüência se números dentro de uma dada faixa e com um dado incremento

sha1sum Prints or checks 160-bit Secure Hash Algorithm 1 (SHA1) checksums

sha224sum Imprime ou checa assinaturas de 224-bit Secure Hash Algorithm

sha256sum Imprime ou checa assinaturas de 256-bit Secure Hash Algorithm

sha384sum Imprime ou checa assinaturas de 384-bit Secure Hash Algorithm

sha512sum Imprime ou checa assinaturas de 3512-bit Secure Hash Algorithm

shred Sobrescreve os arquivos dados repetidamente com padrões complexos, tornando difícil a recuperação dos dados

shuf Embaralha linhas de texto

sleep Pausa pelo tempo determinado

sort	Ordena as linhas de um dado arquivo
split	Divide um arquivo em pedaços, por tamanho ou por número de linhas
stat	Exibe o status de arquivo ou de sistema de arquivos
stdbuf	Roda comandos com oerações de buffering alteradas para seu stream padrão
stty	Configura ou exibe configurações de linhas do terminal
sum	Exibe a assinatura e a contagem de blocos para cada arquivo dado
sync	Esvazia os buffers do sistema de arquivos; isto força a gravação dos blocos alterados no disco
tac	Concatena os arquivos dados em ordem reversa
tail	Imprime as últimas dez linhas (ou o número dado de linhas) de cada arquivo avaliado
tee	Lê da entrada padrão enquanto escreve tanto para saída padrão quando para determinados arquivos
test	Compara valores e verifica tipos de arquivos
timeout	Roda um comando com um limite de tempo
touch	Muda timestamps do arquivo, colocando a data de acesso e modificação do arquivo para a data atual; arquivos que não existem são criados com comprimento zero
tr	Traduz, comprime e remove os caracteres fornecidos pela entrada padrão
true	Não faz nada, com sucesso; sempre sai com status indicando sucesso
truncate	Comprime ou expande um arquivo para o tamanho especificado
tsort	Executa uma ordenação topológica; ele escreve uma lista completamente ordenada de acordo com a ordenação parcial em um dado arquivo
tty	Retorna o nome do arquivo do terminal conectado a entrada padrão
uname	Exibe informações sobre o sistema
unexpand	Converte espaços para tabs
uniq	Descarta todas, exceto uma dentre sucessivas inhas idênticas
unlink	Remove o arquivo
users	Exibe o nome dos usuários atualmente logados
vdir	É o mesmo que **ls -l**
wc	Exibe o número de linhas, palavras e bytes de cada arquivo dado, assim como um total de linhas quando mais de um arquivo é avaliado
who	Exibe quem está logado
whoami	Exibe o nome de usuário associado ao ID de usuário
yes	Repetidamente retorna “y” ou uma dada string até que seja terminado
`libstdbuf.so`	Biblioteca usada por **stdbuf**

6.27. Iana-Etc-2.30

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

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 2.2 MB

6.27.1. Instalação do Iana-Etc

O seguinte comando converte os dados em estado brutos disponibilizados pelo IANA em formatos corretos para os arquivos de dados `/etc/protocols` e `/etc/services`:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

6.27.2. Conteúdo do Iana-Etc

Arquivos instalados: /etc/protocols and /etc/services

Breve descrição

`/etc/protocols`	Descreve os vários protocolos DARPA da Internet que estão disponíveis no subsistema TCP/IP
`/etc/services`	Disponibiliza um mapeamento entre nomes textuais amigáveis para serviços de internet e seus número de portas e tipos de protocolos não expostos

6.28. M4-1.4.17

O pacote M4 contém um processador de macro.

Tempo de Construção: 0.4 SBU
Espaço de disco: 30 MB

6.28.1. Instalação do M4

Prepare o M4 para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.28.2. Conteúdo do M4

Programa instalado: m4

Breve descrição

m4 copia os arquivos dados enquanto expande as macros que eles contém. Essas macros são built-in ou definidas pelo usuário e podem receber qualquer quantidade de argumentos. Além de executar expansão de macro, **m4** tem funções nativas para a inclusão de arquivos, execução de comandos Unix, cálculo de aritmética de inteiros, manipulação de texto de diversas formas, recursividade, etc. O programa **m4** pode ser usado como um front-end para um compilador ou como um processador de macros independente.

6.29. Flex-2.5.38

O pacote Flex serve para gerar programas que reconhecem padrões em textos.

Tempo de Construção: 0.4 SBU
Espaço de disco: 39 MB

6.29.1. Instalação do Flex

Primeiro, evite a execução de três testes de regressão que requerem bison.

```
sed -i -e '/test-bison/d' tests/Makefile.in
```

Prepare o Flex para a compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/flex-2.5.38
```

Compile o pacote:

```
make
```

Para testa o resultado (cerca de 0.5 SBU), execute:

```
make check
```

Instale o pacote:

```
make install
```

Alguns programas ainda não sabem sobre o **flex** e tentam usar seu predecessor, **lex**. Para suportar esses programas, crie um script encapsulador chamado `lex` que chama `flex` em modo de emulação **lex**:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

6.29.2. Conteúdo do Flex

Programas instalados: flex, flex++ (link to flex), and lex
Bibliotecas instaladas: libfl.{a,so} and libfl_pic.{a,so}
Diretórios instalado: /usr/share/doc/flex-2.5.38

Breve descrição

flex Uma ferramenta para gerar programas que reconhecem padrões em textos; ele permite que haja a versatilidade de especificar as regras para encontrar padrões, erradicando a necessidade de desenvolver programas especializados

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

lex Um script que roda **flex** no modo de emulação **lex**

`libfl` A biblioteca `flex`

6.30. Bison-3.0.2

O pacote Bison contém um gerador de analisador (parser generator).

Tempo de Construção: 0.3 SBU
Espaço de disco: 31 MB

6.30.1. Instalação do Bison

Prepare o Bison para a compilação:

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

Compile o pacote:

```
make
```

Para testa o resultado (cerca de 0.5 SBU), execute:

```
make check
```

Instale o pacote:

```
make install
```

6.30.2. Conteúdo do Bison

Programas instalados: bison and yacc
Biblioteca instalada: liby.a
Diretório instalado: /usr/share/bison

Breve descrição

bison Gera, a partir de uma série de regras, um programa 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 programas que ainda executam a chamada **yacc** em vez de **bison**; ele chama **bison** com a opção *-y*

`liby.a` A biblioteca Yacc contém implementações de funções do "Yacc-compatible" `yyerror` e funções `main`; esta biblioteca normalmente não é muito útil, mas o POSIX a exige

6.31. Grep-2.16

O pacote Grep contém programas para procurar dentro de arquivos.

Tempo de Construção: 0.4 SBU
Espaço de disco: 30 MB

6.31.1. Instalação do Grep

Prepare o Grep para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.31.2. Conteúdo do Grep

Programas instalados: egrep, fgrep, e grep

Breve descrição

egrep Exibe linhas de arquivos que satisfazem a um padrão em uma expressão regular estendida

fgrep Exibe linhas que combinam com uma lista de strings fixas

grep Exibe linhas que combinam com uma expressão regular básica

6.32. Readline-6.2

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

Tempo de Construção: 0.1 SBU
Espaço de disco: 17.2 MB

6.32.1. Instalação do Readline

Reinstalar Readline fará com que as bibliotecas antigas sejam movidas para <libraryname>.old. Mesmo que isto normalmente não seja um problema, em alguns casos isso pode causar um bug de vinculação em **ldconfig**. Isso pode ser evitado executando os dois sed seguintes:

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

Aplique um patch para consertar um bug conhecido que foi corrigido:

```
patch -Np1 -i ../readline-6.2-fixes-2.patch
```

Prepare o Readline para a compilação:

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

Compile o pacote:

```
make SHLIB_LIBS=-lncurses
```

O significado das opções do make:

`SHLIB_LIBS=-lncurses`
Esta opção força o Readline a vincular com a biblioteca `libncurses` (de fato, `libncursesw`).

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

Agora mova as bibliotecas dinâmicas para uma localização mais adequada e conserte alguns links simbólicos:

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

Se desejar, instale a documentação:

```
mkdir   -v /usr/share/doc/readline-6.2
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-6.2
```

6.32.2. Conteúdo do Readline

Bibliotecas instaladas: libhistory.{a,so}, and libreadline.{a,so}
Diretórios instalado: /usr/include/readline, /usr/share/readline, /usr/share/doc/readline-6.2

Breve descrição

`libhistory`	Fornece uma interface consistente com o usuário recordando o histórico da linha de comando
`libreadline`	Ajuda a dar consistência à interface de usuário através de programas discretos que precisam disponibilizar uma interface de linha de comando

6.33. Bash-4.2

Este pacote contém o Burn-Again SHell

Tempo de Construção: 1.7 SBU
Espaço de disco: 45 MB

6.33.1. Instalação do Bash

Primeiro, aplique o patch seguinte para consertar vários bugs que foram abordados anteriormente:

```
patch -Np1 -i ../bash-4.2-fixes-12.patch
```

Prepare o Bash para a compilação:

```
./configure --prefix=/usr                      \
            --bindir=/bin                      \
            --htmldir=/usr/share/doc/bash-4.2 \
            --without-bash-malloc              \
            --with-installed-readline
```

O significado das opções do configure:

`--htmldir`
Esta opção designa o diretório dentro do qual a documentação HTML formatada será instalada.

`--with-installed-readline`
Esta opção diz para o Bash usar a biblioteca `readline` que já está instalada no sistema em vez de usar sua própria versão do readline.

Compile o pacote:

```
make
```

Pule para “Instalar o pacote” se voc# não vai rodar a suite de testes.

Para preparar os testes, garanta que o usuário `nobody` pode escrever nas árvores de diretório do código fonte:

```
chown -Rv nobody .
```

Agora, rode o teste como o usuário `nobody`:

```
su nobody -s /bin/bash -c "PATH=$PATH make tests"
```

Instale o pacote:

```
make install
```

Rode o programa recém compilado **bash** (substituindo o que está sendo usado atualmente):

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

Nota

Os parâmetros usados fazem do processo **bash** um shell de login interativo e continua a desabilitar hashing de modo que os novos programas sejam encontrados conforme estejam disponíveis.

6.33.2. Conteúdo do Bash

Programas instalados: bash, bashbug, and sh (link to bash)
Diretório instalado: /usr/share/doc/bash-4.2

Breve descrição

bash Um interpretador de comandos vastamente utilizado;ele executa vários tipos de expansões e substituições em uma dada linha de comando antes de executá-la, fazendo portanto desse interpretador uma ferramenta poderosa

bashbug Um script para ajudar o usuário a compor e enviar emails padronizados reportando bugs relacionados ao **bash**

sh m symlink para o programa **bash**; quando invocado como **sh**, **bash** tenta imitar o comportamento de inicialização de versões históricas do **sh** de maneira tão fiel quanto possível, enquanto mantendo conformidade com o padrão POSIX

6.34. Bc-1.06.95

O pacote Bc contém uma linguagem de processamento numérica de precisão arbitrária.

Tempo de Construção: 0.1 SBU
Espaço de disco: 3 MB

6.34.1. Instalação do Bc

Prepare Bc para compilação:

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

O significado das opções do configure:

`--with-readline`
Esta opção diz ao Bc para usar a biblioteca `readline` que já está instalada no sistema em vez de usar sua própria versão.

Compile o pacote:

```
make
```

Para testar o bc, rode os comandos abaixo.. Há bastante saída, portanto você vai querer redirecionar a saída para um arquivo. Há uma percentagem de testes muita baixa (10 de 12,144) que irão indicar erro de arredondamento no último digito.

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

Instale o pacote:

```
make install
```

6.34.2. Contents of Bc

Programas instalados: bc and dc

Breve descrição

bc é uma calculadora de linha de comando

dc é uma calculadora de linha de comando de entrada polonesa(reversa)

6.35. Libtool-2.4.2

O pacote Libtool contém o script genérico GNU de suporte à biblioteca. Ele esconde a complexidade do uso de bibliotecas compartilhadas em uma interface consistente e portável.

Tempo de Construção: 3.0 SBU
Espaço de disco: 37 MB

6.35.1. Instalação do Libtool

Prepare o Libtool para a compilação:

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

Compile o pacote:

```
make
```

Para testa o resultado (cerca de 3.0 SBU), execute:

```
make check
```

Instale o pacote:

```
make install
```

6.35.2. Conteúdo do Libtool

Programas instalados: libtool e libtoolize
Bibliotecas instaladas: libltdl.{a,so}
Diretórios instalado: /usr/include/libltdl, /usr/share/libtool

Breve descrição

libtool Fornece serviços genéricos de suporte à compilação de bibliotecas

libtoolize Fornece um meio padronizado para adicionar suporte ao **libtool** a um pacote

`libltdl` Esconde as várias dificuldades no desenvolvimento de bibliotecas

6.36. GDBM-1.11

O pacote GDBM contém o GNU Database Manager(Gerenciador de Bancos de dados GNU). Este é um banco de dados no formado de arquivos em disco que armazena chaves/pares de dados em arquivos únicos. Os dados reais de cada registro mantido são indexados por uma chave única, que pode ser encontrada em menos tempo do que se fosse armazenado em um arquivo de texto.

Tempo de Construção: 0.1 SBU
Espaço de disco: 8.5 MB

6.36.1. Instalação do GDBM

Prepare o GDBM para a compilação:

```
./configure --prefix=/usr --enable-libgdbm-compat
```

O significado da opção do configure:

`--enable-libgdbm-compat`
Este switch habilita a compilação da biblioteca de compatibilidade libgdbm, uma vez que alguns pacotes fora do LFS podem requerer as rotinas antigas do DBM que ela dispõe.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.36.2. Conteúdo do GDBM

Programa instalado: testgdbm
Bibliotecas instaladas: libgdbm.{a,so} and libgdbm_compat.{a,so}

Breve descrição

testgdbm Testa e modifica bancos de dados GDBM
`libgdbm` Contém funções para manipular um banco de dados "hashed"

6.37. Inetutils-1.9.2

O pacote Inetutils contém programas para funcionamento básico de rede.

Tempo de Construção: 0.4 SBU
Espaço de disco: 27 MB

6.37.1. Instalação do Inetutils

Crie uma definição para permitir que o programa **ifconfig** construa adequadamente.

```
echo '#define PATH_PROCNET_DEV "/proc/net/dev"' >> ifconfig/system/linux.h
```

Prepare o Inetutils para a compilação:

```
./configure --prefix=/usr  \
    --localstatedir=/var   \
    --disable-logger       \
    --disable-syslogd      \
    --disable-whois        \
    --disable-servers
```

O significado das opções do configure:

`--disable-logger`
Esta opção evita que o Inetutils instale o programa **logger**, que é usado por scripts para passar mensagens para o "System Log Daemon". Não instale isso porque o Util-linux instalou uma versão anteriormente.

`--disable-syslogd`
Esta opção evita que o Inetutils instale o System Log Daemon, que é instalado com o pacote Sysklogd.

`--disable-whois`
Esta opção desabilita a compilação do cliente **whois**, do Inetutils, o qual está desatualizado. Instruções para um cliente **whois** melhor estão no livro BLFS.

`--disable-servers`
Isto desabilita a instalação de vários servidores de rede incluídos como parte do pacote Inetutils. Esses pacotes são avaliados como não adequados em um sistema LFS básico. Alguns são inseguros por natureza e são considerados seguras apenas em redes confiáveis. Note que melhores substituições estão disponíveis para muitos desses servidores.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Mova alguns programas de modo que eles estejam disponíveis se `/usr` não estiver acessível:

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

6.37.2. Conteúdo do Inetutils

Programas instalados: ftp, ifconfig, hostname, ping, ping6, rcp, rexec, rlogin, rsh, talk, telnet, tftp, e traceroute

Breve descrição

ftp	É o programa para o protocolo de transferência de arquivos
ifconfig	Gerencia interfaces de rede
hostname	Exibe ou configura o nome do host
ping	Envia pacotes echo e retorna quando tempo a resposta demora
ping6	Uma versão do **ping** para redes IPv6
rcp	Executa cópia remota de arquivos
rexec	Executa comandos em um ambiente remoto
rlogin	Executa login remoto
rsh	Roda um shell remoto
talk	É usado para conversar com outro usuário
telnet	Uma interface para o protocolo TELNET
tftp	Um programa trivial de transferência de arquivo
traceroute	Traça a rota que seu pacote toma do seu computador para outro computador na rede, mostrando todos os saltos (gateways) pelo caminho

6.38. Perl-5.18.2

O pacote Perl contém a linguagem "Practical Extraction and Report Language".

Tempo de Construção: 6.7 SBU
Espaço de disco: 246 MB

6.38.1. Instalação do Perl

Primeiro crie um arquivo `/etc/hosts` básico para ser referenciado em um dos arquivos de configuração do Perl assim como na suite de testes opcional:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Esta versão do Perl agora compila o módulo Compress::Raw::Zlib. Por padrão Perl usará uma cópia interna da fonte do Zlib para a compilação. Execute o seguinte comando de modo que o Perl usará a biblioteca Zlib instalada no sistema:

```
sed -i -e "s|BUILD_ZLIB\s*= True|BUILD_ZLIB = False|"           \
       -e "s|INCLUDE\s*= ./zlib-src|INCLUDE    = /usr/include|" \
       -e "s|LIB\s*= ./zlib-src|LIB        = /usr/lib|"        \
    cpan/Compress-Raw-Zlib/config.in
```

Para ter controle completo sobre a configuração do Perl, você pode remover as opções “-des” do comando seguinte e escolher a maneira como o pacote é compilado. Alternativamente, use o comando exatamente como está abaixo para usar os padrões que o Perl detecta automaticamente:

```
sh Configure -des -Dprefix=/usr                 \
                  -Dvendorprefix=/usr           \
                  -Dman1dir=/usr/share/man/man1 \
                  -Dman3dir=/usr/share/man/man3 \
                  -Dpager="/usr/bin/less -isR"  \
                  -Duseshrplib
```

O significado das opções do configure:

`-Dvendorprefix=/usr`
Isso garante que **perl** saiba como dizer aos pacotes onde eles devem instalar seus módulos perl.

`-Dpager="/usr/bin/less -isR"`
Isso corrige um erro na maneira como **perldoc** invoca o programa **less**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Uma vez que o Groff ainda não está instalado, o **Configure** acha que nós não queremos páginas de manual para o Perl. Aplicar esses parâmetros sobrepõe essa decisão.

`-Duseshrplib`
Compile uma libperl compartilhada necessária para alguns módulos perl.

Compile o pacote:

```
make
```

Para testa os resultados (aproximadamente 2.5 SBU), execute:

```
make -k test
```

Instale o pacote:

```
make install
```

6.38.2. Conteúdo do Perl

Programas instalados: a2p, c2ph, config_data, corelist, cpan, cpan2dist, cpanp, cpanp-run-perl, enc2xs, find2perl, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.18.2 (link to perl), perlbug, perldoc, perlivp, perlthanks (link to perlbug), piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (link to s2p), pstruct (link to c2ph), ptar, ptardiff, ptargrep, s2p, shasum, splain, xsubpp, e zipdetails
Bibliotecas instaladas: Várias centenas que não podem ser listadas aqui
Diretório instalado: /usr/lib/perl5

Breve descrição

a2p Traduz awk para Perl

c2ph Exibe as estruturas C como se fossem geradas a partir de **cc -g -S**

config_data Consulta ou altera a configuração de módulos Perl

corelist Um frontend de linha de comando para Module::CoreList

cpan Interage com o "Comprehensive Perl Archive Network" (CPAN) da linha de comando

cpan2dist O criador de distribuição CPANPLUS

cpanp O lançador CPANPLUS

cpanp-run-perl Script Perl que é usado para habilitar a descarga do buffer de saída após cada escrita em processos 'spawned'

enc2xs Compila uma extensão Perl para o módulo Encode tanto do Unicode Character Mappings quanto dos arquivos Tcl Encoding

find2perl Traduz comandos **find** para Perl

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 shell para examinar módulos Perl instalados, e pode até criar tarball de módulos instalados

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 única linguagem muito poderosa

perl5.18.2 Um hard link para **perl**

perlbug Usado para gerar relatórios de bug sobre o Perl ou seus módulos e os envia por e-mail

perldoc Apresenta uma parte da documentação em formato .pod encontrada nos diretórios do Perl ou em um script Perl e a exibe

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 a serem enviadas por email para desenvolvedores Perl
piconv	Uma versão Perl do conversor de codificação de caracteres **iconv**
pl2pm	Uma ferramenta grosseira para converter arquivos Perl4 `.pl` para módulos Perl5 `.pm`
pod2html	Converte arquivos do formato pod para o formato HTML
pod2latex	Converte arquivos do formato pod para o formato LaTeX
pod2man	Converte dados pod para entrada formatada *roff
pod2text	Converte dados pod para texto fASCII formatado
pod2usage	Exibe mensagens de utilização de documentos pod embutidos em arquivos
podchecker	Verifica a sintaxe de arquivos de documentação no formato pod
podselect	Exibe seções selecionadas de documentações pod
prove	Ferramenta de linha de comando para executar testes no múdulo Test::Harness.
psed	Uma versão Perl do editor de stream **sed**
pstruct	Exibe as estruturas C como se fossem geradas a partir de **cc -g -S**
ptar	Um programa similar ao **tar** escrito em Perl
ptardiff	Um programa que compara um arquivo extraído com um que não foi extraído
ptargrep	Um programa Perl que aplica verificação de padrões ao conteúdo de arquivos em um arquivo tar
s2p	Traduz scripts **sed** para Perl
shasum	Exibe e checa somas SHA
splain	É usado para forçar diagnósticos de avisos detalhado no Perl
xsubpp	Converte código Perl XS em código C
zipdetails	Exibe detalhes sobre a estrutura interna de um arquivo Zip

6.39. Autoconf-2.69

O pacote Autoconf contém programas para produzir “shell scripts” que podem configurar automaticamente o código fonte.

Tempo de Construção: 4.5 SBU
Espaço de disco: 17.1 MB

6.39.1. Instalação do Autoconf

Prepare o Autoconf para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Isso leva muito tempo, cerca de 4.7 SBUs. Adicionalmente, 6 testes que usam Automake são pulados. Para cobertura completa do teste, Autoconf pode ser re-testado depois que o Automake tenha sido instalado.

Instale o pacote:

```
make install
```

6.39.2. Conteúdo do Autoconf

Programas instalados: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate, e ifnames
Diretório instalado: /usr/share/autoconf

Breve descrição

autoconf Produz scripts shell que configuram automaticamente o código fonte de pacotes para adaptar a vários tipos de sistemas baseados no Unix. Os scripts de configuração que ele produz são independentes —rexecutá-los não requer o programa **autoconf**.

autoheader Uma ferramenta para criar arquivos template de declarações C *#define* para uso do configure

autom4te Um wrapper para o processador de macro M4

autoreconf Automaticamente executa **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize**, e **libtoolize** na ordem correta para salvar tempo quando mudanças são feitos nos arquivos template **autoconf** e **automake**

autoscan Ajuda a criar um arquivo `configure.in` fpara um pacote de software; ele examina os arquivos fonte na árvore de diretório, procurando por problema de portabilidade comuns, e cria um arquivo `configure.scan` que serve preliminarmente como um arquivo `configure.in` para o pacote

autoupdate Modifica um arquivo `configure.in` que ainda chama macros **autoconf** através de seus nomes antigos para que use os nomes de macros atuais

ifnames Ajuda enquanto escrevendo arquivos `configure.in` para um pacote de software; ele imprime os identificadores que o pacote usa em condicionais do preprocessador C. Se um pacote já foi

configurado para ter certa portabilidade, este programa pode ajudar a determinar o que o **configure** precisa checar. Ele também pode preencher uma falha no arquivo `configure.in` gerada pelo **autoscan**

6.40. Automake-1.14.1

O pacote Automake contém programas para gerar Makefiles para uso com Autoconf.

Tempo de Construção: menos de 0.1 SBU (cerca de 12 SBU com testes)
Espaço de disco: 100 MB

6.40.1. Instalação do Automake

Prepare o Automake para a compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.14.1
```

Compile o pacote:

```
make
```

Há alguns testes que ligam incorretamente para a versão errada da biblioteca flex, então nós temporariamente resolveremos isso. Além disso, usando a opção -j4 aumenta a velocidade do teste, mesmo em sistemas com apenas um processador devido a atrados internos nos testes individuais. Teste o resultado, execute:

```
sed -i "s:./configure:LEXLIB=/usr/lib/libfl.a &:" t/lex-{clean,depend}-cxx.sh
make -j4 check
```

Instale o pacote:

```
make install
```

6.40.2. Conteúdo do Automake

Programas instalados: acinstall, aclocal, aclocal-1.14, automake, automake-1.14, compile, config.guess, config.sub, depcomp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, e ylwrap
Diretórios instalado: /usr/share/aclocal-1.14, /usr/share/automake-1.14, /usr/share/doc/automake-1.14.1

Breve descrição

acinstall Um script que instala arquivos M4 aclocal-style M4.

aclocal Gera arquivos `aclocal.m4` baseados no conteúdo dos arquivos `configure.in`

aclocal-1.14 Um hard link para **aclocal**

automake Uma ferramenta para gerar automaticamente arquivos `Makefile.in` dos arquivos `Makefile.am`. Para criar todos os arquivos `Makefile.in` fpara um pacote, rode este programa no diretório mais acima. Escaneando o arquivo `configure.in` ele automaticamente encontra cada arquivo `Makefile.am` apropriado e gera o arquivo `Makefile.in` correspondente

automake-1.14 Um hard link para **automake**

compile Um wrapper para compiladores

config.guess Um script que tenta adivinhar o trio canônico para uma dada compilação, host ou arquitetura

config.sub Um script de validação de configuração

depcomp Um script para compilar um programa de modo que informações de dependência é gerada em adição a saída desejada

install-sh	Um script que instala um programa, script, ou arquivo de dados
mdate-sh	Um script que imprime o horário de modificação de um arquivo ou diretório
missing	Um script que atua como um stub comum para
mkinstalldirs	Um script que cria uma árvore de diretórios
py-compile	Compila um programa Python
ylwrap	Um encapsulador para **lex** e **yacc**

6.41. Diffutils-3.3

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

Tempo de Construção: 0.5 SBU
Espaço de disco: 25 MB

6.41.1. Instalação do Diffutils

Primeiro conserte um arquivo para que os arquivos de locale sejam instalados:

```
sed -i 's:= @mkdir_p@:= /bin/mkdir -p:' po/Makefile.in.in
```

Prepare o Diffutils para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.41.2. Conteúdo do Diffutils

Programas instalados: cmp, diff, diff3, and sdiff

Breve descrição

cmp Compara dois arquivos e reporta se eles se diferem e que bytes
diff Compara dois arquivos ou diretórios e reporta que linhas nos arquivos eles diferem
diff3 Compara três linhas de arquivos por linha
sdiff Junta dois arquivos e interativamente exibe os resultados

6.42. Gawk-4.1.0

O pacote Gawkcontém programas para manipular arquivos de texto.

Tempo de Construção: 0.2 SBU
Espaço de disco: 30 MB

6.42.1. Instalação do Gawk

Prepare o Gawk para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Se desejar, instale a documentação:

```
mkdir -v /usr/share/doc/gawk-4.1.0
cp    -v doc/{awkforai.txt,*.{eps,pdf,jpg}} /usr/share/doc/gawk-4.1.0
```

6.42.2. Conteúdo do Gawk

Programas instalados: awk (link to gawk), gawk, gawk-4.1.0, e igawk
Bibliotecas instaladas: filefuncs.so, fnmatch.so, fork.so, inplace.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so, testext.so, e time.so
Diretórios instalado: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk, /usr/share/doc/gawk-4.1.0

Breve descrição

awk Um link para **gawk**

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

gawk-4.1.0 Um hard link para **gawk**

igawk Dá ao **gawk** a habilidade de incluir arquivos

6.43. Findutils-4.4.2

O pacote Findutils contém programas para procurar arquivos. Esses programas são disponibilizados para procurar recursivamente dentro de uma árvore de diretórios e para criar, manter e buscar um banco de dados (geralmente mais rápido que o find recursivo, não confiável de o banco de dados não foi atualizado recentemente).

Tempo de Construção: 0.4 SBU
Espaço de disco: 29 MB

6.43.1. Instalação do Findutils

Prepare o Findutils para a compilação:

```
./configure --prefix=/usr --localstatedir=/var/lib/locate
```

O significado das opções do configure:

`--localstatedir`
This option changes the location of the **locate** database to be in `/var/lib/locate`, which is FHS-compliant.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Alguns dos scripts no pacote LFS-Bootscripts dependem do **find**. Como `/usr` pode não estar disponível durante os primeiros estágios da inicialização, esses programas precisam estar na partição raiz. O script **updatedb** também precisa ser modificado para corrigir um patch explícito:

```
mv -v /usr/bin/find /bin
sed -i 's/find:=${BINDIR}/find:=\/bin/' /usr/bin/updatedb
```

6.43.2. Conteúdo do Findutils

Programas instalados: bigram, code, find, frcode, locate, oldfind, updatedb, e xargs

Breve descrição

bigram Foi anteriormente usado para produzir bancos de dados **locate**

code Foi anteriormente usado para produzir bancos de dados **locate**; é o ancestral de **frcode**.

find Busca em árvores de diretórios arquivos que satisfazem critérios especificados

frcode É chamado pelo **updatedb** para comprimir a lista de nomes de arquivos; ele usa "front-compression", reduzindo o tamanho do banco de dados em um fator de quatro ou cinco.

locate Busca em um banco de dados de nomes de arquivos e retorna os nomes que contém a string dada ou que satisfazem o padrão fornecido

oldfind Versão antiga do find, usando um algoritmo diferente

updatedb Atualiza o banco de dados **locate**; ele escaneia todo o sistema de arquivos (incluindo outros sistemas de arquivos montados, a menos que dito o contrário) e coloca todos os nomes de arquivos que ele encontra em um banco de dados

xargs Pode ser usado para aplicar um dado comando a uma lista de arquivos.

6.44. Gettext-0.18.3.2

O pacote Gettext contém utilitários para internacionalização e localização. Eles permitem que programas sejam compilados com NLS (Native Language Support), habilitando-os a emitir mensagens na língua nativa do usuário.

Tempo de Construção: 2.3 SBU
Espaço de disco: 199 MB

6.44.1. Instalação do Gettext

Prepare o Gettext para a compilação:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gettext-0.18.3.2
```

Compile o pacote:

```
make
```

Para testar os resultados (isto leva muito tempo, cerca de 0.3 SBU), execute:

```
make check
```

Instale o pacote:

```
make install
```

6.44.2. Conteúdo do Gettext

Programas instalados: autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin, e xgettext
Bibliotecas instaladas: libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so}, libgettextsrc.so, e preloadable_libintl.so
Diretórios instalado: /usr/lib/gettext, /usr/share/doc/gettext-0.18.3.2, /usr/share/gettext

Breve descrição

autopoint Copia arquivos de infraestrutura padrão do Gettext em um pacote fonte

config.charset Exibe uma tabela de pseudônimos de codificação de caracteres dependente do sistema

config.rpath Exibe um conjunto de variáveis dependente do sistema, descrevendo como configurar o caminho de busca em tempo de execução das bibliotecas compartilhadas em um executável

envsubst Substitui variáveis de ambiente em strings no formato do shell

gettext Traduz uma mensagem em linguagem natural para a língua do usuário buscando a tradução em um catálogo de mensagens

gettext.sh Primariamente serve como uma bibliotecas de funções do shell para o gettext

gettextize Copia todos os arquivos Gettext padrão dentro de um dado diretório de um pacote para começar a internacionalizá-lo

hostname Exibe o nome do computador da rede em vários formatos

msgattrib	Filtra as mensagens de um catálogo de tradução de acordo com os seus atributos e manipula os atributos
msgcat	Concatena e funde os arquivos `.po` fornecidos
msgcmp	Compara dois arquivos `.po` para checar que ambos contém o mesmo conjunto de strings msgid
msgcomm	Encontra as mensagens que são comuns aos arquivos `.po`
msgconv	Converte um catálogo traduzido 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 satisfazem um dado parâmetro ou que pertencem a algum arquivo fonte fornecido
msginit	Cria um novo arquivo `.po`, inicializando a meta informação com valores do ambiente do usuário
msgmerge	Combina duas traduções cruas em um único arquivo
msgunfmt	Descompila um catálogo de mensagem binário em um texto traduzido cru
msguniq	Unifica traduções duplicadas em um catálogo de tradução
ngettext	Exibe linguagem de tradução nativa de uma mensagem textual cuja gramática depende de um número
recode-sr-latin	Recodifica texto sérvio do cirílico para o alfabeto latino
xgettext	Extrai as linhas que podem ser traduzidas de um arquivo para fazer o primeiro modelo de tradução
`libasprintf`	define a classe *autosprintf*, que faz as rotinas de saída formatada do C utilizáveis em programas C++, para uso com <*string*> e <*iostream*>
`libgettextlib`	uma biblioteca privada contendo rotinas comuns usadas por vários programas Gettext; elas não são para uso geral
`libgettextpo`	Usado para escrever programas especializados que processam arquivos `.po`; esta biblioteca é usada com as aplicações padrão que vem com o Gettext (tais como **msgcomm**, **msgcmp**, **msgattrib**, e **msgen**) não serão suficiente
`libgettextsrc`	Uma biblioteca privada que contem rotinas comuns usadas por vários programas Gettext; elas não são para uso geral
`preloadable_libintl`	Uma biblioteca, para ser usada por LD_PRELOAD que ajuda `libintl` a registrar mensagens não traduzidas.

6.45. Groff-1.22.2

O pacote Groff contém programas para processamento e formatação de texto.

Tempo de Construção: 0.5 SBU
Espaço de disco: 83 MB

6.45.1. Instalação do Groff

Groff espera que a variável de ambiente PAGE contenha o tamanho de papel padrão. Para usuários nos Estados Unidos, *PAGE=letter* é apropriado. Nos demais locais, *PAGE=A4* pode ser mais adequado. Mesmo que o tamanho do papel padrão seja configurado durante a compilação, isso pode ser sobrescrito com "A4" ou "letter" no arquivo `/etc/papersize`.

Prepare o Groff para a compilação:

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

Compile o pacote:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

Alguns programas de documentação, como **xman**, não funcionarão propriamente sem os seguintes symlinks:

```
ln -sv eqn /usr/bin/geqn
ln -sv tbl /usr/bin/gtbl
```

6.45.2. Conteúdo do Groff

Programas instalados: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, geqn (link to eqn), grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (link to tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit, e troff
Diretórios instalado: /usr/lib/groff, /usr/share/doc/groff-1.22.2, /usr/share/groff

Breve descrição

addftinfo Lê um arquivo de fonte troff e adiciona algumas informações sobre medidas de fontes que são usadas pelo sistema **groff**

afmtodit Cria um arquivo de fonte para uso com **groff** e **grops**

chem Processador Groff para produzir diagramas de estruturas químicas

eqn Compila descrições de equações embutidas em arquivos de entrada troff em comandos que são entendidos por **troff**

eqn2graph Converte um troff EQN (equação) em uma imagem

gdiffmk	Marca diferenças entre arquivos groff/nroff/troff
geqn	Um link para **eqn**
grap2graph	Converte um diagrama "grap" em uma imagem bitmap
grn	Um processador **groff** para arquivos gremlin
grodvi	Um driver para **groff** que produz o formato TeX dvi
groff	Um front-end para o sistema de formatação de documentos groff; normalmente, ele roda o programa **troff** e um pós-processador apropriado para o dispositivo selecionado
groffer	Exibe arquivos groff e páginas de manual no X e em terminais tty
grog	Lê arquivos e advinha quais das opções **groff** `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s`, e `-t` são necessárias para imprimir arquivos, e retorna o comando **groff** cincluindo essas opções
grolbp	É um driver **groff** para impressoras Canon CAPSL (séries de impressoras a laser LBP-4 e LBP-8)
grolj4	É um driver para **groff** que produz saída no formato PCL5 adequado para uma impressora HP LaserJet 4
grops	Traduz a saída do GNU **troff** para PostScript
grotty	Traduz a saída do GNU **troff** em uma forma adequada para dispositivos tipo máquina de escrever
gtbl	Um link para **tbl**
hpftodit	Cria um arquivo fonte para uso com **groff -Tlj4** a partir de um arquivo com fontes com métrica marcadas no formato HP
indxbib	Cria um índice invertido para o banco de dados bibliográfico com um arquivo especificado para uso com **refer**, **lookbib**, e **lkbib**
lkbib	Vasculha bancos de dados bibliográficos por referências que contém chaves específicas e reporta quaisquer referências encontradas
lookbib	Exibe uma linha de comando na saída de erros padrão (a não ser que a entrada padrão não seja um terminal), lê da entrada padrão uma linha contendo um conjunto de caracteres, procura por referências contendo estes caracteres em bases de dados bibliográficos em um arquivo especificado, exibe na saída padrão as referências encontradas e repete este processo até o final da entrada fornecida
mmroff	Um processador simples para **groff**
neqn	Formata equações para a saída "American Standard Code for Information Interchange" (ASCII)
nroff	Um script que emula o comando **nroff** usando **groff**
pdfroff	Cria documentos pdf usando groff
pfbtops	Traduz uma fonte PostScript no formato `.pfb` para ASCII
pic	Compila descrições de imagens embutidas em arquivos de entrada troff ou TeX em comando entendidos pelo TeX ou **troff**
pic2graph	Converte um diagrama PIC em uma imagem
post-grohtml	Traduz a saída do GNU **troff** para HTML
preconv	Converte a 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 as linhas entre *.[* e *.]* que são interpretadas como citações, e linhas entre *.R1* e *.R2* são interpretadas como comandos para ditar como as citações serão processadas
roff2dvi	Transforma arquivos roff para o formato DVI
roff2html	Transforma arquivos roff para o formato HTML
roff2pdf	Transforma arquivos roff para o formato PDF
roff2ps	Transforma arquivos roff para o formato ps
roff2text	Transforma arquivos roff em arquivos de texto
roff2x	Transforma arquivos roff para outros formatos
soelim	Lê arquivos e substitui linhas da forma *.so arquivo* pelo conteúdo do *arquivo* mencionado
tbl	Compila descrições de equações embutidas em arquivos de entrada troff em comandos que são entendidos por **troff**
tfmtodit	Cria um arquivo de fonte para uso com **groff -Tdvi**
troff	É altamente compatível com o **troff** do Unix; usualmente deve ser invocado usando o comando **groff**, o qual também rodara um preprocessador e um pós-processador na ordem apropriada e com as opções apropriadas

6.46. Xz-5.0.5

O pacote Xz contém programas para compressão e descompressão de arquivos. Ele fornece recursos para lidar com os novos formatos de compressão lzma e xz. Comprimir arquivos de texto com **xz** gera uma texa de compressão melhor do que os tradicionais comandos **gzip** ou **bzip2**.

Tempo de Construção: 0.3 SBU
Espaço de disco: 18 MB

6.46.1. Instalação do Xz

Prepare o Xz para compilação com:

```
./configure --prefix=/usr --docdir=/usr/share/doc/xz-5.0.5
```

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote e certifique-se que todos os arquivos essenciais estejam nos diretórios corretos:

```
make install
mv -v   /usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat} /bin
mv -v /usr/lib/liblzma.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/liblzma.so) /usr/lib/liblzma.so
```

6.46.2. Conteúdo do Xz

Programas instalados: lzcat (link to xz), lzcmp (link to xzdiff), lzdiff (link to xzdiff), lzegrep (link to xzgrep), lzfgrep (link to xzgrep), lzgrep (link to xzgrep), lzless (link to xzless), lzma (link to xz), lzmadec, lzmainfo, lzmore (link to xzmore), unlzma (link to xz), unxz, (link to xz), xz, xzcat (link to xz), xzcmp (link to xzdiff), xzdec, xzdiff, xzegrep (link to xzgrep), xzfgrep (link to xzgrep), xzgrep, xzless, xzmore
Bibliotecas instaladas: liblzma.{a,so}
Diretórios instalado: /usr/include/lzma and /usr/share/doc/xz-5.0.5

Breve descrição

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 informações armazenadas em cabeçalhos de arquivos comprimidos 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 o algoritmo de compressão de dados em cadeira Lempel-Ziv-Markov sem perdas com ordação de blocos.

6.47. GRUB-2.00

O pacte GRUB contém o GRand Unified Bootloader.

Tempo de Construção: 0.7 SBU
Espaço de disco: 112 MB

6.47.1. Instalação do GRUB

Conserte uma incompatibilidade entre este pacote e Glibc-2.19:

```
sed -i -e '/gets is a/d' grub-core/gnulib/stdio.in.h
```

Prepare o GRUB para a compilação:

```
./configure --prefix=/usr          \
            --sbindir=/sbin        \
            --sysconfdir=/etc      \
            --disable-grub-emu-usb \
            --disable-efiemu       \
            --disable-werror
```

A opção --disable-werror permite que a compilação seja completada com warnings introduzidos por versões do flex mais recentes. O outro --disable switches minimiza o que é compilado desabilitando características e programas de teste não necessários para o LFS.

Compile o pacote:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

O uso do GRUB para fazer o seu sistema LFS inicializável será discutido no Seção 8.4, “Usando o GRUB para Configurar o Processo de Boot”.

6.47.2. Conteúdo do GRUB

Programas instalados: grub-bios-setup, grub-editenv, grub-fstest, grub-install, grub-kbdcomp, 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-script-check, grub-set-default, grub-sparc64-setup
Diretórios instalado: /usr/lib/grub, /etc/grub.d, /usr/share/grub, /boot/grub

Breve descrição

grub-bios-setup É um programa de ajuda para grub-install
grub-editenv Uma ferramenta para editar o bloco de ambiente
grub-fstest Ferramenta para depurar o driver do sistema de arquivos

grub-install	Instala o GRUB no seu drive
grub-kbdcomp	Script que converte um layout xkb layout em um que pode ser reconhecido pelo GRUB
grub-menulst2cfg	Converte um antigo arquivo `menu.lst` do GRUB em um `grub.cfg` para uso com GRUB 2
grub-mkconfig	Gera um arquivo grub config
grub-mkimage	Gera uma imagem inicializável do GRUB
grub-mklayout	Gera um arquivo de layout de teclado do GRUB
grub-mknetdir	Prepara um diretório do GRUB para boot pela rede
grub-mkpasswd-pbkdf2	Gera uma senha PBKDF2 encriptada para uso com o menu de inicialização
grub-mkrelpath	Faz o caminho do sistema relativo a seu diretório raiz
grub-mkrescue	Faz uma imagem do GRUB inicializável 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 informações de dispositivo em um dado caminho ou dispositivo
grub-reboot	Define a entrada de boot padrão do GRUB apenas para a próxima inicialização
grub-script-check	Verifica arquivos de configuração do GRUB por erros de sintaxe
grub-set-default	Define a entrada de boot padrão para o GRUB
grub-sparc64-setup	É um programa auxiliar para grub-setup

6.48. Less-458

O pacote Less contém um visualizador de arquivos de texto.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 3.6 MB

6.48.1. Instalação do Less

Prepare o Less para a compilação:

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

O significado das opções do configure:

`--sysconfdir=/etc`
Esta opção diz aos programas criados pelo pacote para procurarem pelos arquivos de configuração em `/etc`.

Compile o pacote:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make install
```

6.48.2. Conteúdo do Less

Programas instalados: less, lessecho, e lesskey

Breve descrição

less Um visualizador de arquivos ou paginador; exibe o conteúdo do arquivo especificado, com recursos de rolagem pelo usuário, busca por strings e salto para marcações

lessecho Necessário para expandir meta-caracteres, tais como * e ?, iem nomes de arquivos em sistemas Unix

lesskey Usado para especificar as teclas de atalho para **less**

6.49. Gzip-1.6

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

Tempo de Construção: 0.2 SBU
Espaço de disco: 19.7 MB

6.49.1. Instalação do Gzip

Prepare o Gzip para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Move alguns programas que não precisam estar o sistema de arquivos raiz:

```
mv -v /bin/{gzexe,uncompress,zcmp,zdiff,zegrep} /usr/bin
mv -v /bin/{zfgrep,zforce,zgrep,zless,zmore,znew} /usr/bin
```

6.49.2. Conteúdo do Gzip

Programas instalados: gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore, e znew

Breve descrição

gunzip Descomprime arquivos compactados com gzip

gzexe Cria arquivos executáveis que se descomprimem automaticamente

gzip Comprime um dado arquivo usando a codificação Lempel-Ziv (LZ77)

uncompress Descomprime arquivos comprimidos

zcat Descomprime um dado arquivo que foi comprimido com gzip para a saída padrão

zcmp Executa **cmp** em arquivos compactados com gzip

zdiff Executa **diff** em arquivos compactados com gzip

zegrep Executa **egrep** em arquivos compactados com gzip

zfgrep Executa **fgrep** em arquivos compactados com gzip

zforce Força uma extensão `.gz` em todos os arquivos fornecido, de modo que o **gzip** não irá comprimi-los novamente;isso pode ser útil quando nomes de arquivos foram truncados durante a transferência de arquivos

zgrep	Executa **grep** em arquivos compactados com gzip
zless	Executa **less** em arquivos compactados com gzip
zmore	Executa **more** em arquivos compactados com gzip
znew	Recomprime arquivos com o formato **compress** para o formato **gzip** — `.Z` para `.gz`

6.50. IPRoute2-3.12.0

O pacote IPRoute2 contém programas para funcionamento de rede baseada em IPV4 básico e avançado.

Tempo de Construção: 0.1 SBU
Espaço de disco: 7.3 MB

6.50.1. Instalação do IPRoute2

O binário **arpd** incluído neste pacote é dependente do Berkeley DB. Devido ao foto do **arpd** não ser um requerimento muito comum em um sistema Linux básico, remova a dependência no Berkeley DB aplicando os comandos abaixo. Se o binário **arpd** for necessário, instruções para compilar o Berkeley DB podem ser encontradas no livro BLFS no link *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db*.

```
sed -i '/^TARGETS/s@arpd@@g' misc/Makefile
sed -i /ARPD/d Makefile
sed -i 's/arpd.8//' man/man8/Makefile
```

Compile o pacote:

```
make DESTDIR=
```

O significado das opções do make:

`DESTDIR=`
Isso garante que os binários do IPRoute2 serão instalados no diretório correto. Por padrão, *`DESTDIR`* é configurado para `/usr`.

Este pacote vem com uma suite de testes, mas devido a suposições que ele faz, não é possível rodar a suite de testes de maneira confiável a partir do ambiente chroot. Se você deseja rodar esses testes depois de inicializar em seu novo sistema LFS, certifique-se de selecionar o suporte `/proc/config.gz` CONFIG_IKCONFIG_PROC ("General setup" -> "Enable access to .config through /proc/config.gz") no seu kernel, então rode 'make alltests' do subdiretório `testsuite/`.

Instale o pacote:

```
make DESTDIR=              \
     MANDIR=/usr/share/man \
     DOCDIR=/usr/share/doc/iproute2-3.12.0 install
```

6.50.2. Conteúdo do IPRoute2

Programas instalados: bridge, ctstat (link to lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (link to lnstat), ss, e tc
Diretórios instalado: /etc/iproute2, /lib/tc, /usr/share/doc/iproute2-3.12.0, /usr/lib/tc

Breve descrição

bridge Configura pontes de redes

ctstat Utilitário para status de conexão

genl

ifcfg Um script shell encapsulador para o comando **ip**. Note que ele requer os programas **arping** e **rdisk** do pacote iputils encontrado em *http://www.skbuff.net/iputils/*.

ifstat Mostra as estatísticas da interface, incluindo a quantidade de pacotes transmitidos e recebidos pela interface

ip O executável principal. Ele tem várias funções diferentes:
ip link *<device>* permite aos usuários olharem para o estado de um dispositivo e fazer mudanças
ip addr permite que usuários vejam endereços e suas propriedades, adicionem novos endereços, e deletem velhos
ip neighbor permite que os usuários olhem emperramentos vizinhos e suas propriedades, adicionem entradas vizinhas novas, e deletem as velhas
ip rule permite que os usuários vejam as políticas da distribuição e as mudem
ip route permite que os usuários vejam a tabela de distribuição e para modifiquem suas regras
ip tunnel permite que os usuários vejam e modifiquem os túneis do IP e suas propriedades
ip maddr permite que os usuários vejam e modifiquem os endereços multicast e suas propriedades
ip mroute permite que os usuários ajustem, mudem, ou deletem a distribuição do multicast
ip monitor permite usuários monitorem continuamente o estado dos dispositivos, dos endereços e das rotas

lnstat Fornece as estatísticas da rede Linux. É uma versão mais completa para o antigo programa **rtstat**

nstat Mostra estatísticas de rede

routef Um complemento do **ip route**. Esvazia as tabelas de distribuição

routel Um complemento do **ip route**. Lista as tabelas de distribuição

rtacct Exibe o conteúdo de `/proc/net/rt_acct`

rtmon Utilitário de monitoramento de rota

rtpr Converte a saída do **ip -o** em um formato legível

rtstat Utilitário de status da rota

ss Similar ao comando **netstat**; exibe conexões ativas

tc Controlador de Tráfego Executável; (Traffic Controlling Executable); implementação para Quality Of Service (QOS) e Class Of Service (COS)
tc qdisc permite que os usuários definam a disciplina da fila
tc class permite que os usuários definam as classes baseadas na programação da disciplina da fila
tc estimator permite que usuários estime o fluxo de rede
tc filter permite que os usuários definam o filtro de pacotes do QOS/COS
tc policy permite que os usuários definam as políticas de QOS/COS

6.51. Kbd-2.0.1

O pacote Kbd contém arquivos com tabelas de caracteres, fontes de console, e utilitários para teclado.

Tempo de Construção: 0.1 SBU
Espaço de disco: 20 MB

6.51.1. Instalação do Kbd

O comportamento das teclas 'Backspace' e 'Delete' não é consistente nos mapas de teclados no pacote Kbd. O patch seguinte conserta este problema para keymaps i386:

```
patch -Np1 -i ../kbd-2.0.1-backspace-1.patch
```

Depois de aplicar o patch, a tecla Backspace gera o caracter com código 127, e a tecla Delete gera uma seqüência bem conhecida de escape.

Remova o programa redundante **resizecons** (ele requer que a biblioteca morta svgalib disponibilize os arquivos de modo de vídeo - para uso normal **setfont** dimensiona o console adequadamente) juntamente com sua página de manual.

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

Prepare o Kbd para a compilação:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr --disable-vlock
```

O significado das opções do configure:

`--disable-vlock`
Esta opção evita que o utilitário vlock seja compilado, uma vez que ele requer a biblioteca PAM, que não está disponível no ambiente chroot.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Nota

Para algumas línguas (e.g., Belarusian) o pacote Kbd não dispõe de um mapa de caracteres útil onde o keymap regular “by” supõe a codificação ISO-8859-5, e o mapa de caracteres CP1251 é usado. Usuários de tais línguas tem que baixar um mapa de caracteres funcional separadamente.

Se desejar, instale a documentação:

```
mkdir -v       /usr/share/doc/kbd-2.0.1
cp -R -v docs/doc/* /usr/share/doc/kbd-2.0.1
```

6.51.2. Conteúdo do Kbd

Programas instalados: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (link to psfxtable), psfgettable (link to psfxtable), psfstriptable (link to psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, showconsolefont, showkey, unicode_start, e unicode_stop

Diretórios instalado: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/unimaps

Breve descrição

chvt	Modifica o terminal virtual de primeiro plano
deallocvt	Libera terminais virtuais não usados
dumpkeys	Exibe as tabelas de conversão do teclado
fgconsole	Exibe o número do terminal virtual ativo
getkeycodes	Exibe a tabela de mapeamento scancode-para-keycode do kernel
kbdinfo	Obtém informação sobre o status de um console
kbd_mode	Reporta ou configura o modo do teclado
kbdrate	Configura as taxas de repetição e atraso do teclado
loadkeys	Carrega as tabelas de tradução do teclado
loadunimap	Carrega a tabela de mapeamento unicode-para-fonte do kernel
mapscrn	Um programa obsoleto que carrega a tabela de saída de caractere definida pelo usuário dentro do driver do console; Isto é feito atualmente pelo **setfont**
openvt	Inicia um programa em um novo terminal virtual (VT).
psfaddtable	Um link para **psfxtable**
psfgettable	Um link para **psfxtable**
psfstriptable	Um link para **psfxtable**
psfxtable	Manipula tabelas de caracteres Unicode para fontes do console
setfont	Muda as fontes do Enhanced Graphic Adapter (EGA) e Video Graphics Array (VGA) no console
setkeycodes	Carrega as entradas da tabela de mapeamento scancode-para-keycode do kernel; isto é útil se houver teclas incomuns no teclado
setleds	Configura as flags do teclado e os Light Emitting Diodes (LEDs)
setmetamode	Define o funcionamento da tecla alt (meta) do teclado
showconsolefont	Exibe a atual fonte EGA/VGA de tela do console
showkey	Mostra os scancodes, os keycodes, e os códigos ASCII das teclas pressionadas no teclado
unicode_start	Põe o teclado e o console no modo UNICODE. Não use este programa a menos que o seu arquivo de mapa de teclado esteja na codificação ISO-8859-1. Para outras codificações, este programa produz resultados incorretos.
unicode_stop	Retira o teclado e o console do modo Unicode

6.52. Kmod-16

O pacote Kmod contém bibliotecas e utilitários para carregar módulos do kernel

Tempo de Construção: 0.1 SBU
Espaço de disco: 34 MB

6.52.1. Instalação do Kmod

Prepare o Kmod para a compilação:

```
./configure --prefix=/usr          \
            --bindir=/bin          \
            --sysconfdir=/etc      \
            --with-rootlibdir=/lib \
            --disable-manpages     \
            --with-xz              \
            --with-zlib
```

O significado das opções do configure:

`--with-xz, --with-zlib`
Essas opções habilitam o Kmod a manipular módulos comprimidos do kernel.

`--disable-manpages`
Esta opção evita que as páginas de manual sejam construídas, uma vez que elas dependem de libxslt, que não está disponível no ambiente chroot.

`--with-rootlibdir=/lib`
Isso garante que diferentes arquivos relacionados a bibliotecas sejam colocados nos diretórios corretos.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote, páginas de manual, e cria os links simbólicos para compatibilidade com Module-Init-Tools (o pacote que previamente manipulava módulos do kernel Linux). Além disso faça com que toas as bibliotecas estejam nos diretórios corretos:

```
make install
make -C man install

for target in depmod insmod modinfo modprobe rmmod; do
  ln -sv ../bin/kmod /sbin/$target
done

ln -sv kmod /bin/lsmod
```

6.52.2. Conteúdo do Kmod

Programas instalados: depmod (link to kmod), insmod (link to kmod), kmod, lsmod (link to kmod), modinfo (link to kmod), modprobe (link to kmod), e rmmod (link to kmod)
Bibliotecas instaladas: libkmod.so

Breve descrição

depmod Cria um arquivo de dependência baseado nos símbolos que ele encontra no conjunto de módulos existente; este arquivo de dependência é usado pelo **modprobe** para carregar automaticamente os módulos necessários

insmod Instala um módulo carregavel no kernel que está rodando

kmod Carrega e descarrega módulos do kernel

lsmod Lista módulos do kernel atualmente carregados

modinfo Examina um arquivo objeto associado a um módulo do kernel e exibe qualquer informação que ele pode colher

modprobe Usa um arquivo de dependência, criado por **depmod**, para carregar automaticamente módulos relevantes

rmmod Descarrega módulos do kernel corrente

`libkmod` Esta biblioteca é usada por outros programas para carregar e descarregar módulos do kernel

6.53. Libpipeline-1.2.6

O pacote Libpipeline contém bibliotecas para manipular pipelines de sub-processos de uma maneira flexível e conveniente.

Tempo de Construção: 0.2 SBU
Espaço de disco: 7.4 MB

6.53.1. Instalação do Libpipeline

Prepare o Libpipeline para a compilação:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure --prefix=/usr
```

O significado das opções do configure:

PKG_CONFIG_PATH
Use pkg-config para obter a localização dos meta-dados da biblioteca de testes construída em Seção 5.14, “Check-0.9.12”.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.53.2. Conteúdo do Libpipeline

Bibliotecas instaladas: libpipeline.so

Breve descrição

`libpipeline` Esta biblioteca é usada para construir com segurança 'pipelines' entre subprocessos

6.54. Make-4.0

O pacote Make contém um programa para compilação de pacotes.

Tempo de Construção: 0.4 SBU
Espaço de disco: 11.3 MB

6.54.1. Instalação do Make

Prepare o Make para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.54.2. Conteúdo do Make

Programa instalado: make

Breve descrição

make Determina automaticamente que partes de um pacote necessitam ser (re)compiladas e emite então os comandos apropriados

6.55. Patch-2.7.1

O pacote Patch contém um programa para modificar ou criar arquivos aplicando um arquivo “patch” tipicamente criado pelo programa **diff**.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 3.4 MB

6.55.1. Instalação do Patch

Prepare o Patch para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.55.2. Conteúdo do Patch

Programa instalado: patch

Breve descrição

patch Modifica arquivos de acordo com arquivos patch. Um arquivo patch normalmente é uma lista de diferenças criada com o programa **diff**. Aplicando essas diferenças ao arquivo original, **patch** cria a versão "patched".

6.56. Sysklogd-1.5

O pacote Sysklogd contém programas para gravação das mensagens de log do sistema, como aquelas reportadas pelo kernel quando coisas incomuns acontecem.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 0.6 MB

6.56.1. Instalação do Sysklogd

Compile o pacote:

```
make
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make BINDIR=/sbin install
```

6.56.2. Configurando o Sysklogd

Crie um novo arquivo `/etc/syslog.conf` rodando:

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

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

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

6.56.3. Conteúdo do Sysklogd

Programas instalados: klogd and syslogd

Breve descrição

klogd Um serviço de sistema (daemon) que intercepta e registra mensagens do kernel

syslogd Registra mensagens que programas do sistema oferecem para serem registradas. Casa mensagem registrada contém pelo menos um registro de tempo e hostname, e normalmente o nome do programa também, mas isso depende do quão confiável o daemon de registro é

6.57. Sysvinit-2.88dsf

O pacote Sysvinit contém programas para controlar a inicialização, a execução e a finalização do sistema.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 1.4 MB

6.57.1. Instalação do Sysvinit

Primeiro, aplique um patch que remove vários programas instalados por outros pacotes, esclarece uma mensage, e conserta um warning de um compilador:

```
patch -Np1 -i ../sysvinit-2.88dsf-consolidated-1.patch
```

Compile o pacote:

```
make -C src
```

Este pacote não contém uma suite de testes.

Instale o pacote:

```
make -C src install
```

6.57.2. Conteúdo do Sysvinit

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

Breve descrição

bootlogd	Registra mensagens de boot em um arquivo de log
fstab-decode	Roda um comando com argumentos codificados para fstab
halt	Normalmente invoca **shutdown** com a opção *-h*, exceto quando já está em run-level 0, então ele diz ao kernel para desligar o sistema; nota-se no arquivo `/var/log/wtmp` que o sistema está sendo desligado
init	O primeiro processo a ser iniciado depois que o kernel inicializa o hardware e que toma conta do processo de inicialização e inicia todos os processos que é instruído
killall5	Envia um sinal para todos os processos, exceto para processo em sua própria seção de modo que não matará o shell rodando o script que o invocou
poweroff	Diz ao kernel para desativar o sistema e desligar o computador (veja **halt**)
reboot	Diz ao kernel para reiniciar o sistema (veja **halt**)
runlevel	Reporta o run-level atual e anterior, conforme o último run-level em `/var/run/utmp`
shutdown	Desliga o sistema de maneira segura, sinalizando a todos os processos e notificando todos os usuários logados
telinit	Diz ao **init** para que run-level mudar

6.58. Tar-1.27.1

O pacote Tar contém um programa para arquivamento.

Tempo de Construção: 2.4 SBU
Espaço de disco: 34 MB

6.58.1. Instalação do Tar

Adiciona um programa que gera uma página de manual para o tar a partir do código fonte:

```
patch -Np1 -i ../tar-1.27.1-manpage-1.patch
```

Prepare o Tar para a compilação:

```
FORCE_UNSAFE_CONFIGURE=1  \
./configure --prefix=/usr \
            --bindir=/bin
```

O significado das opções do configure:

`FORCE_UNSAFE_CONFIGURE=1`
This forces the test for `mknod` to be run as root. É geralmente considerado perigoso rodar este teste como usuário root, mas como está sendo executado em um sistema que foi apenas parcialmente construído, está OK ignorar isso.

Compile o pacote:

```
make
```

Para testa o resultado (cerca de 1 SBU), execute:

```
make check
```

Instale o pacote:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.27.1
```

Finalmente, gere a página de manual e coloque-a em sua localização correta:

```
perl tarman > /usr/share/man/man1/tar.1
```

6.58.2. Conteúdo do Tar

Programas instalados: rmt and tar
Diretório instalado: /usr/share/doc/tar-1.27.1

Breve descrição

rmt Manipula remotamente um drive de fica magnética através de uma conexão interprocessada

tar Cria, extrai arquivos de, e lista o conteúdo dos pacotes de arquivos, conhecidos também como tarballs

6.59. Texinfo-5.2

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

Tempo de Construção: 0.6 SBU
Espaço de disco: 101 MB

6.59.1. Instalação do Texinfo

Prepare o Texinfo para a compilação:

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

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

Opcionalmente, instale os componentes que pertencem a uma instalação do TeX:

```
make TEXMF=/usr/share/texmf install-tex
```

O significado do parâmetro do make:

TEXMF=/usr/share/texmf
A variável TEXMF armazena a posição da raiz da árvore do TeX se, por exemplo, o pacote TeX for instalado mais tarde.

O sistema de documentação Info utiliza um arquivo texto puro para armazenar sua lista de entradas de menu. Este arquivo está localizado em /usr/share/info/dir. Infelizmente, devido a problemas ocasionais nos makefiles de vários pacotes, ele pode às vezes sair da sincronização com as páginas do info instaladas no sistema. Se o arquivo /usr/share/info/dir fprecisar ser recriado, os seguintes comandos opcionais realizam esta tarefa:

```
cd /usr/share/info
rm -v dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.59.2. Conteúdo do Texinfo

Programas instalados: info, infokey, install-info, makeinfo (link to texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf, e texindex
Diretório instalado: /usr/share/texinfo

Breve descrição

info	Usado para ler páginas info que são parecidas com páginas man, mas geralmente vão muito mais a fundo do que simples explicações das opções de linha de comando. Por exemplo, compare **man bison** e **info bison**.
infokey	Compila um arquivo fonte contento customizações info em um formato binário
install-info	Usado para instalar páginas de manual; atualiza entradas no arquivo de índice **info**
makeinfo	Converte documentos-fontes Texinfo para os formatos info, texto puro e HTML
pdftexi2dvi	Usado para formatar o documento Texinfo dado em um PDF (Portable Document Format)
pod2texi	Converte Pod para formato Texinfo
texi2any	Traduz documentos fonte Texinfo para vários outros formatos
texi2dvi	Usado para formatar o documento Texinfo dado em um arquivo independente de dispositivo que pode ser impresso
texi2pdf	Usado para formatar o documento Texinfo dado em um PDF (Portable Document Format)
texindex	Usado para ordenar arquivos de índice Texinfo

6.60. Udev-208 (Extraído de systemd-208)

O pacote Udev contém programas para a criação dinâmica de nós de dispositivos. O desenvolvimento do udev foi fundido com systemd, mas boa parte do systemd é incompatível com LFS. Aqui nós compilamos e instalamos só os arquivos udev necessários.

Tempo de Construção: 0.1 SBU
Espaço de disco: 29 MB

6.60.1. Instalação do Udev

Nota

Este pacote é um pouco diferente de outros pacotes. O pacote inicial que é extraído é `systemd-208.tar.xz` ainda que a aplicação que nós estamos instalando seja udev. Após mudar para o diretório systemd, siga as instruções abaixo.

O tarball udev-lfs contém arquivos específicos para o LFS usados para compilar o Udev. Descompacte-o dentro do diretório fonte do systemd:

```
tar -xvf ../udev-lfs-208-3.tar.bz2
```

Crie dois links simbólicos para arquivos de cabeçalho e defina uma variável de ambiente para usar propriamente Seção 5.33, “Util-linux-2.24.1”.

```
ln -svf /tools/include/blkid /usr/include
ln -svf /tools/include/uuid  /usr/include
export LD_LIBRARY_PATH=/tools/lib
```

Construa o pacote:

```
make -f udev-lfs-208-3/Makefile.lfs
```

Instale o pacote:

```
make -f udev-lfs-208-3/Makefile.lfs install
```

Cuidado

Há vários locais dentro do código fonte de systemd que têm caminhos de diretórios explícitos embutidos. Por exemplo, a versão binária do caminho do banco de dados de hardware e o nome do arquivo usado em tempo de execução, `/etc/udev/hwdb.bin`, não podem ser modificados sem mudanças explícitas no código fonte.

Agora inicialize o banco de dados de hardware:

```
build/udevadm hwdb --update
```

Finalmente configure as regras de rede persistentes do udev. Esta tarefa será explicada em detalhes em Seção 7.2.1, “Criando nomes estáveis para interfaces de rede”. Note que os sistemas de arquivos `/sys` e `/proc` devem estar montados no ambiente chroot como explicado no início deste capítulo para que o seguinte script funcione.

```
bash udev-lfs-208-3/init-net-rules.sh
```

Faça uma limpeza:

```
rm -fv /usr/include/{uuid,blkid}
unset LD_LIBRARY_PATH
```

6.60.2. Conteúdo do Udev

Programas instalados: accelerometer, ata_id, cdrom_id, collect, mtd_probe, scsi_id, v4l_id, udevadm, e udevd
Bibliotecas instaladas: libudev.so
Diretórios instalado: /etc/udev, /lib/udev, /lib/firmware, /usr/share/doc/udev

Breve descrição

ata_id Disponibiliza ao Udev uma única string e informação adicional (uuid, label) para um drive ATA

cdrom_id Disponibiliza ao Udev a capacidade de reconhecer drive de CD-ROM ou DVD-ROM

collect Dada uma ID para o uevent atual e uma lista de IDs (para todos uevent alvo), registra a ID atual e indica se todas as IDsalvo foram registradas ou não

scsi_id Disponibiliza ao Udev um único identificador SCSI baseado nos dados retornados do envio de um comando SCSI INQUIRY para o dispositivo especificado

udevadm Ferramenta genérica de administração do udev: controla o daemon udev, disponibiliza informações do banco de dados Udev, monitora uevents, espera que uevents terminem, testa configuração do Udev, e aciona uevents para um dado dispositivo

udevd Um daemon que escuta a uevents no socket netlink, cria dispositivos e roda os programas de configuração externos em resposta a esses uevents

`libudev` Uma biblioteca de interface para informações de dispositivos udev

`/etc/udev` Contém arquivos de configuração Udev, permissões de dispositivos e regras para nomear dispositivos

6.61. Util-linux-2.24.1

O pacote Util-linux contém uma variedade de aplicativos utilitários. Entre eles estão ferramentas para manipulação de sistemas de arquivos, consoles, partições e mensagens.

Tempo de Construção: 0.6 SBU
Espaço de disco: 89 MB

6.61.1. Notas sobre conformidade com o padrão FHS

O FHS recomenda usar o diretório `/var/lib/hwclock` em vez do diretório usual `/etc` como localização do arquivo `adjtime`. To make the **hwclock** program FHS-compliant, run the following:

```
sed -i -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \
          $(grep -rl '/etc/adjtime' .)

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

6.61.2. Instalação do Util-linux

Prepare Util-linux para a compilação:

```
./configure
```

Compile o pacote:

```
make
```

Se desejar, rode a suite de teste como usuário diferente do root:

Atenção

Rodar a suite de teste como root pode causar danos ao seu sistema. Para fazê-lo, a opção CONFIG_SCSI_DEBUG para o kernel tem que estar disponível no sistema em execução, e deve ser constuída como módulo. Construí-lo dentro do kernel irá prevenir a inicialização. Para cobertura completa, outros pacotes do BFLS tem que ser intalados. Caso deseje, esse teste pode ser executado após reiniciar no seu sistema LFS completo e em execução:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

Nota

Dois testes, last/ipv6 e last/last, falham no ambiente chroot devido a "DNS resolver" não estar ativo ainda. Se os testes forem novamente executados após a inicialização, eles passarão.

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

Instale o pacote:

```
make install
```

6.61.3. Conteúdo do Util-linux

Programas instalados: addpart, agetty, blkdiscard, blkid, blockdev, cal, cfdisk, chcpu, chrt, col, colcrt, colrm, column, ctrlaltdel, cytune, delpart, dmesg, eject, fallocate, fdformat, fdisk, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hexdump, hwclock, i386, ionice, ipcmk, ipcrm, ipcs, isosize, kill, last, lastb (link to last), ldattach, linux32, linux64, logger, look, losetup, lsblk, lscpu, lslocks, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, partx, pg, pivot_root, prlimit, raw, readprofile, rename, renice, resizepart, rev, rtcwake, script, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff (link to swapon), swapon, switch_root, tailf, taskset, ul, umount, unshare, utmpdump, uuidd, uuidgen, wall, wdctl, whereis, wipefs, e x86_64

Bibliotecas instaladas: libblkid.{a,so}, libmount.{a,so}, libuuid.{a,so}

Diretórios instalado: /usr/include/blkid, /usr/include/libmount, /usr/include/uuid, /usr/share/doc/util-linux/getopt, /var/lib/hwclock

Breve descrição

addpart	Informa o kernel Linux sobre novas partições
agetty	Abre uma porta tty, pede um nome de login e invoca o programa **login**
blkdiscard	Descarta setores em um dispositivo
blkid	Um utilitário de linha de comando para localizar e imprimir atributos de blocos de dispositivos
blockdev	Permite chamar dispositivos de controle de I/O de bloco (block device ioctls) na linha de comando
cal	Exibe um calendário simples
cfdisk	Manipulador da tabela de partição de um dado dispositivo
chcpu	Modifica o estado de CPUs
chrt	Manipula os atributos de um processo em tempo real
col	Filtra LFs (line feeds) reversos
colcrt	Filtra a saída **nroff** para terminais que não tem algumas capacidades, tais como overstriking e half-lines
colrm	Filtra as colunas dadas
column	Formata um dado arquivo em múltiplas colunas
ctrlaltdel	Configura a função da combinação de teclas CTRL+ALT+DEL (resetar soft ou hard)
cytune	Ajusta os parâmetros para o driver Cyclades
delpart	pede ao kernel Linux para remover uma partição
dmesg	Exibe as mensagens de boot do kernel
eject	Ejeta mídia removível
fallocate	Pré-aloca espaço para um arquivo
fdformat	Formata um disquete em baixo nível
fdisk	Manipulador da tabela de partição de um dado dispositivo
findfs	Encontra um sistema de arquivos pelo label ou pelo Universally Unique Identifier (UUID)

findmnt É uma interface de linha de comando para a biblioteca libmount para trabalho com arquivos mountinfo, fstab e mtab

flock Adquire uma trava de arquivo e então executa um comando o a trava ativada

fsck É usado para verificar e, opcionalmente, reparar sistemas de arquivos

fsck.cramfs Executa uma verificação de consistência no sistema de arquivos Cramfs em um dado dispositivo

fsck.minix Executa uma verificação de consistência no sistema de arquivos minix em um dado dispositivo

fsfreeze É um wrapper muito simples sobre operações de driver de kernel ioctl FIFREEZE/FITHAW

fstrim Descarta blocos não usados em um sistema de arquivos montado

getopt Analisa opções de comandos em uma linha de comando

hexdump Exibe o arquivo em formato hexadecimal ou em outro formato dado

hwclock Exibe e configura o relógio de hardware (também chamado de RTC ou relógio do BIOS).

i386 Um link simbólico para setarch

ionice Verifica ou configura para um programa a prioridade e classe de 'io scheduling'

ipcmk Cria vários recursos IPC

ipcrm Remove um recurso especificado de uma comunicação inter-Process (IPC)

ipcs Fornece informação de status do IPC

isosize Exibe o tamanho de um sistema de arquivos iso9660

kill Envia sinais para processos

last Mostra o ultimo usuário de fez logged in (e out), buscando no arquivo `/var/log/wtmp`; também mostra inicializações do sistema, desligamentos, e mudanças de run-level

lastb Mostra as tentativas de login que falharam, conforme registrado em `/var/log/btmp`

ldattach Anexa uma disciplina de linha para uma linha serial

linux32 Um link simbólico para setarch

linux64 Um link simbólico para setarch

logger Adiciona a mensagem no log do sistema

look Exibe linhas que começam com a dada string

losetup Configura e controla dispositivos loop

lsblk Lista informações sobre blocos de dispositivos em formato de árvore, todos ou selecionados.

lscpu Exibe informação sobre arquitetura da CPU

lslocks Lista travas locais de sistema

mcookie Gera cookies mágicos (números 128-bit hexadecimais aleatórios) para o **xauth**

mesg Controla se outros usuários podem enviar mensagens para o terminal do usuário atual

mkfs Cria um sistema de arquivos em um dispositivo (geralmente uma partição do disco)

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 um dado dispositivo ou arquivo para ser usado como área swap
more	Um filtro para paginação de textos uma tela de cada vez
mount	Anexa o sistema de arquivos em um dado dispositivo a um diretório especificado na árvore de arquivos
mountpoint	Verifica de o diretório é um ponto de montagem
namei	Mostra os links simbólicos nos caminhos dados
nsenter	Roda um programa com namespaces de outros processos
partx	Avisa ao kernel sobre a presença e número de partições no disco
pg	Mostra um arquivo texto uma tela por vez
pivot_root	Faz do sistema de arquivos dados o novo sistema de arquivos raiz para o processo atual
prlimit	Verifica e configura os limites de recursos de um processo
raw	Usado para ligar um dispositivo Linux de caractere cru a um dispositivo de bloco
readprofile	Lê informações de perfil do kernel
rename	Renomeia arquivos, substituindo uma dada string por outra
renice	Altera a prioridade um um processo em execução
resizepart	Pede ao kernel Linux para redimensionar uma partição
rev	Inverte as linhas de um arquivo
rtcwake	Usado para fazer um sistema entrar em 'sleep state' até a hora especificada
script	Cria uma impressão da sessão de terminal
scriptreplay	Executa typescripts usando informações de temporização
setarch	Muda a arquitetura reportada em um novo ambiente de programa e configura flags de personalidade
setsid	Roda o programa em uma nova seção
setterm	Configura atributos do terminal
sfdisk	Manipulador da tabela de partição do disco
sulogin	Permite que `root` faça login; é normalmente invocado pelo **init** quando o sistema entra em modo 'single user'
swaplabel	Permite modificar swaparea UUID e label
swapoff	Desabilita dispositivos e arquivos para paginação e armazenamento temporário
swapon	Habilita dispositivos e arquivos para paginação e armazenamento temporário e lista os dispositivos e arquivos atualmente em uso
switch_root	Muda para outro sistema de arquivos tornando-o raiz da árvore montada
tailf	Acompanha o crescimento de um arquivo log. Exibe as últimas 10 linhas de um arquivo log, então continua exibindo quaisquer novas entradas no arquivo log conforme são criadas
taskset	Retorna ou define a afinidade de um processo com a CPU
ul	Um filtro para traduzir sublinhados em seqüências de escape indicando sublinha para o terminal em uso
umount	Desconecta um sistema de arquivos da árvore de arquivos do sistema

unshare	Roda um programa com alguns namespaces não compartilhados do pai
utmpdump	Exibe o conteúdo do arquivo de login dado em um formato mais amigável
uuidd	Um daemon usado pela biblioteca UUID para gerar UUIDs baseadas na data de forma garantidamente segura
uuidgen	Cria novas UUIDs. Cada nova UUID pode ser considerada única entre doas UUIDs criadas, no sistema local em em outros sistemas, no passado e no futuro
wall	Exibe o conteúdo de um arquivo ou, por padrão, sua entrada padrão, nos terminais de todos os usuários atualmente logados
wdctl	Mostra o status do watchdog de hardware
whereis	Reporta a localizção de um binário, fonte e página de manual para o comando dado
wipefs	Limpa uma assinatura de sistema de arquivos de um dispositivo
x86_64	Um link simbólico para setarch
`libblkid`	Contém rotinas para identificação de dispositivos e extração de token
`libmount`	Contém rotinas para montagem e desmontagem de blocos de dispositivos
`libuuid`	Contém rotinas para gerar identificadores únicos para objetos que podem ser acessíveis além do sistema local

6.62. Man-DB-2.6.6

O pacote Man-DB contém programas para encontrar e visualizar páginas de manuais.

Tempo de Construção: 0.5 SBU
Espaço de disco: 27 MB

6.62.1. Instalação do Man-DB

Prepare o Man-DB para a compilação:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.6.6 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap
```

O significado das opções do configure:

`--disable-setuid`
Isso impede que o comando **man** use setuid para o usuário `man`.

`--with-...`
Esses três parâmetros são usados para configurar alguns programas padrão. **lynx** é um navegador web baseado em texto (veja BLFS para instruções de instalação), **vgrind** converte fontes de programa para formato de entrada do Groff, e **grap** é útil para configurar gráficos em documentos Groff. Os programas **vgrind** e **grap** normalmente não não necessários para visualizar as páginas de manual. Eles não não parte do LFS ou BLFS, mas você deve ser capaz de instalá-los após terminar o LFS se você quiser.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make check
```

Instale o pacote:

```
make install
```

6.62.2. Páginas de Manual que não estão em inglês

A tabela seguinte mostra o conjunto de caracteres que Man-DB supõe que as páginas de manual instaladas em `/usr/share/man/<ll>` estarão codificadas. Em adição, o Man-DB determina corretamente se páginas de manual instaladas naquele diretório estão codificadas com UTF-8.

Tabela 6.1. Codificação de caracteres esperada para páginas de manual antigas de 8-bit

Linguagem (código)	Codificação	Linguagem (código)	Codificação
Danish (da)	ISO-8859-1	Croatian (hr)	ISO-8859-2

Linguagem (código)	Codificação	Linguagem (código)	Codificação
German (de)	ISO-8859-1	Hungarian (hu)	ISO-8859-2
English (en)	ISO-8859-1	Japanese (ja)	EUC-JP
Spanish (es)	ISO-8859-1	Korean (ko)	EUC-KR
Estonian (et)	ISO-8859-1	Lithuanian (lt)	ISO-8859-13
Finnish (fi)	ISO-8859-1	Latvian (lv)	ISO-8859-13
French (fr)	ISO-8859-1	Macedonian (mk)	ISO-8859-5
Irish (ga)	ISO-8859-1	Polish (pl)	ISO-8859-2
Galician (gl)	ISO-8859-1	Romanian (ro)	ISO-8859-2
Indonesian (id)	ISO-8859-1	Russian (ru)	KOI8-R
Icelandic (is)	ISO-8859-1	Slovak (sk)	ISO-8859-2
Italian (it)	ISO-8859-1	Slovenian (sl)	ISO-8859-2
Norwegian Bokmal (nb)	ISO-8859-1	Serbian Latin (sr@latin)	ISO-8859-2
Dutch (nl)	ISO-8859-1	Serbian (sr)	ISO-8859-5
Norwegian Nynorsk (nn)	ISO-8859-1	Turkish (tr)	ISO-8859-9
Norwegian (no)	ISO-8859-1	Ukrainian (uk)	KOI8-U
Portuguese (pt)	ISO-8859-1	Vietnamese (vi)	TCVN5712-1
Swedish (sv)	ISO-8859-1	Simplified Chinese (zh_CN)	GBK
Belarusian (be)	CP1251	Simplified Chinese, Singapore (zh_SG)	GBK
Bulgarian (bg)	CP1251	Traditional Chinese, Hong Kong (zh_HK)	BIG5HKSCS
Czech (cs)	ISO-8859-2	Traditional Chinese (zh_TW)	BIG5
Greek (el)	ISO-8859-7		

Nota

Páginas de manual em linguagens que não estão nesta lista não são suportadas.

6.62.3. Conteúdo do Man-DB

Programas instalados: accessdb, apropos (link to whatis), catman, lexgrog, man, mandb, manpath, whatis, e zsoelim
Bibliotecas instaladas: libman.so, libmandb.so
Diretórios instalado: /usr/lib/man-db, /usr/libexec/man-db, /usr/share/doc/man-db-2.6.6

Breve descrição

accessdb Descarrega o conteúdo do banco de dados **whatis** em formato legível para humanos

apropos Busca no banco de dados **whatis** e exibe as descrições curtas dos comandos de sistema que contém a dada string

catman Cria ou atualiza páginas de manual pré-formatadas

lexgrog Exibe em uma linha informações sobre uma dada página de manual

man Formata e exibe a página de manual solicitada

mandb Cria ou atualiza o banco de dados **whatis**

manpath Exibe o conteúdo de $MANPATH ou (se $MANPATH não estiver configurado) um caminho de busca adequado baseado nas configurações em man.conf e no ambiente do usuário

whatis Busca no banco de dados **whatis** e exibe uma breve descrição dos comandos do sistema que contém a palavra chave dada como uma palavra separada

zsoelim Lê arquivos e substitui linhas da forma *.so arquivo* pelo conteúdo do *arquivo* mencionado

`libman` Contém suporte em tempo de execução para o **man**

`libmandb` Contém suporte em tempo de execução para o **man**

6.63. Vim-7.4

O pacote Vim contém um editor de textos poderoso.

Tempo de Construção: 1.4 SBU
Espaço de disco: 121 MB

Alternativas ao Vim

Se você preferir algum outro editor de texto—omo o Emacs, Joe ou Nano—por favor consulte *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html* para instruções de instalação sugeridas.

6.63.1. Instalação do Vim

Primeiro, mude a localização padrão do arquivo de configuração `vimrc` para `/etc`:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Prepare Vim para a compilação:

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

O significado das opções do configure:

`--enable-multibyte`
Este parâmetro habilita suporte para edição de arquivos em codificação de caracteres multibyte. Isto é necessário quando usando um locale com a definição de caracteres multibyte. Este parâmetro é útil também para permitir a edição de arquivos de texto criados em distribuições Linux como a Fedora que utiliza a UTF-8 como conjunto de caracteres padrão.

Compile o pacote:

```
make
```

Teste o resultado, execute:

```
make test
```

Entretanto, este conjunto de testes gera saída com muitos dados binários na tela, que pode causar problemas com os ajustes atuais do terminal. Isto pode ser resolvido redirecionando a saída para um arquivo de log. Um teste bem sucedido resultará nas palavras "ALL DONE" quando completo.

Instale o pacote:

```
make install
```

Muitos usuários estão acostumados a utilizar **vi** em vez do **vim**. Para permitir a execução do **vim** quando os usuários habitualmente digitarem **vi**, crie um symlink para ambos o binário e a página de manual nas línguas disponibilizadas:

```
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 está instalada em `/usr/share/vim`. O symlink a seguir permite que a documentação seja acessada via `/usr/share/doc/vim-7.4`, tornando-o consistente com a localização da documentação de outros pacotes:

```
ln -sv ../vim/vim74/doc /usr/share/doc/vim-7.4
```

Se um sistema de janelas X (X Window System) vier a ser instalado no sistema LFS, pode ser necessária a recompilação do Vim após a instalação do X. O Vim tem uma versão GUI do editor que requer o X e algumas bibliotecas adicionais para ser instalado. Para mais informações sobre este processo, procure a documentação do Vim e a página de instalação do Vim no livro BLFS em *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.63.2. Configurando o Vim

Por padrão, **vim** é executado em um modo incompatível com o vi. Isso pode ser novo para usuários que usaram outros editores no passado. A configuração “nocompatible” é incluída abaixo para destacar o fato de que um novo modo está sendo usado. Lembra também àqueles que mudariam para o modo “compatible” que este deve ser o primeiro ajuste no arquivo de configuração. Isto é necessário porque muda outras configurações, e substituições devem vir após esta definição. Crie um arquivo de configuração padrão **vim** rodando o seguinte:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

O *`set nocompatible`* faz com que o **vim** comporte-se de um modo mais útil (o padrão) do que no modo compatível com vi. Remova o “no” para manter o velho comportamento **vi**. O *`set backspace=2`* permite retroceder sobre quebra de linhas, autoindentações e no começo da inserção. O parâmetro *`syntax on`* habilita o destaque de sintaxe do vim. Finalmente, o operador *if* com a instrução *`set background=dark`* corrige a suposição do **vim** sobre a cor do fundo em alguns emuladores de terminal. Isto dá o destaque a um esquema de cores melhor para o uso no fundo preto destes programas.

A documentação para outras opções disponíveis pode ser obtida executando o seguinte comando:

```
vim -c ':options'
```

Nota

Por padrão, Vim apenas instala arquivos spell para a língua inglesa. Para instalar arquivos spell para sua língua preferida, baixe os arquivos `*.spl` e, opcionalmente, os arquivos `*.sug` para sua língua e docificação de caracteres em *ftp://ftp.vim.org/pub/vim/runtime/spell/* e salve-os em `/usr/share/vim/vim74/spell/`.

Para usar esses arquivos spell, alguma configuração em `/etc/vimrc` é necessária, e.g.:

```
set spelllang=en,ru
set spell
```

Para mais informação, veja o arquivo README apropriado localizado na URL acima.

6.63.3. Conteúdo do Vim

Programas instalados: ex (link to vim), rview (link to vim), rvim (link to vim), vi (link to vim), view (link to vim), vim, vimdiff (link to vim), vimtutor, e xxd
Diretório instalado: /usr/share/vim

Breve descrição

ex Inicia o **vim** no modo ex

rview É uma versão restrita do **view**; nenhum comando shell pode ser iniciado e o **view** não pode ser suspenso

rvim É uma versão restrita do **vim**; nenhum comando shell pode ser iniciado e o **vim** não pode ser suspenso

vi Um link para **vim**

view Inicia o **vim** em modo read-only (apenas leitura)

vim É o editor

vimdiff Edita duas ou três versões de um arquivo com o **vim** e exibe as diferenças

vimtutor Ensina o básico sobre as teclas chave e os comandos do **vim**

xxd Cria um arquivo decimal do arquivo dado; também pode fazer o contrário, então pode ser usado para patching binários

6.64. Sobre os Símbolos de depuração

Muitos programas e bibliotecas são, por padrão, compilados com símbolos de depuração incluídos (com a opção **gcc**'s *-g*). Isso significa que quando depurando um programa ou biblioteca que foi compilado com informação de depuração incluída, o depurador pode prover não apenas endereços de memória, mas também os nomes das rotinas e variáveis.

Entretanto, a inclusão desses símbolos de depuração aumenta o programa ou biblioteca significativamente. A seguir um exemplo da quantidade de espaço que esses símbolos ocupam:

- Um binário **bash** com os símbolos de depuração: 1200 KB
- Um binário **bash** m os 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

Tamanho pode variar dependendo de qual compilador de biblioteca C foram usados, mas quando comparando programas com e sem os símbolos de depuração, a diferença será geralmente um fator entre dois e cinco.

Como muitos usuários nunca usarão um depurador em seus programas, muito espaço de disco pode ser recuperado removendo esses símbolos. A próxima seção mostra como remover todos os símbolos de depuração dos programas e das bibliotecas.

6.65. Stripping (Esvaziando) Novamente

Se o usuário não for um programador e não pretende fazer depuração no software de seu sistema, o tamanho do sistema pode diminuir aproximadamente 90 MB removendo os símbolos de debbug dos binários e das bibliotecas. Isto não causa nenhum inconveniente, exceto não poder fazer depuração dos programas.

A maioria das pessoas que usam o comando abaixo não têm nenhuma dificuldade. Entretanto, é fácil cometer um erro de digitação e tornar o sistema novo inutilizável, assim, antes de executar o comando **strip**, é uma boa idéia fazer um backup do sistema LFS em seu estado atual.

Antes de executa-lo tome cuidado especial em verificar se nenhuns dos binários esteja em execução. Se não tiver certeza do usuário incorporado no comando chroot usado na Seção 6.4, “Entrando no ambiente Chroot,” fprimeiro saia do chroot:

```
logout
```

Então reentre com:

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

Agora podemos aplicar o strip nos binários e nas bibliotecas com segurança:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
    -exec /tools/bin/strip --strip-debug '{}' ';'
```

Um grande número de arquivos serão reportados como tendo formato não reconhecido. Esses avisos podem ser ignorados com segurança. Esses avisam indicam que aqueles arquivos são scripts em vez de binários.

6.66. Limpando

Finalmente, limpe alguns arquivos extra deixados pela execução de testes:

```
rm -rf /tmp/*
```

Daqui em diante, quando reentrar o ambiente chroot após a saída, use o comando chroot modificado a seguir:

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

A razão para isso é que os programas em `/tools` não são mais necessários. Uma vez que eles não são mais necessários você pode deletar o diretório `/tools` se desejar.

Nota

Remover `/tools` irá remover também cópias temporárias de Tcl, Expect, e DejaGNU que foram usados para rodar os testes com toolchain. Se você precisar desses programas posteriormente, eles precisão ser recompilados e reinstalados. O livro BLFS tem instruções para isso (veja *http://www.linuxfromscratch.org/blfs/*).

```
rm -rf /tools
```

Se os sistemas de arquivos virtuais do kernel foram desmontados, tanto manualmente como por reinicialização, certifique-se que os sistemas de arquivos virtuais do kernel sejam montados quando reentrar com chroot. Esse processo foi explicado em Seção 6.2.2, “Montando e Povoando /dev” e Seção 6.2.3, “Montando os Sistemas de Arquivos Virtuais do Kernel”.

Capítulo 7. Configurando os scripts de inicialização do sistema (Bootscripts)

7.1. Introdução

Este capítulo discute arquivos de configuração e scripts de inicialização. Primeiro, os arquivos de configuração gerais necessários para configurar a rede são apresentados.

- Seção 7.2, “Configuração Geral de Rede.”
- Seção 7.3, “Customizando o Arquivo /etc/hosts.”

Segundo, questões que afetam a configuração adequada de dispositivos são discutidas.

- Seção 7.4, “Manipulando dispositivos e módulos em um Sistema LFS.”
- Seção 7.5, “Criando Symlinks para Dispositivos.”

A seção seguinte detalha como instalar e configurar os scripts do sistema LFS necessários durante o processo de inicialização. A maioria destes scripts trabalharão sem modificações, mas alguns vão exigir arquivos adicionais de configuração porque tratam de informações que dependem do hardware.

Os scripts de inicialização ao estilo System-V são utilizados neste livro porque são largamente aceitos e relativamente simples. Para opções adicionais, uma dica detalha a instalação dos scripts de inicialização ao estilo BSD está disponível em *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. Procurar nas listas de discussão do LFS por“depinit”, “upstart”, ou “systemd” também oferecerá informações adicionais.

Se utilizar um estilo alternativo de script init, pule esta seção.

Uma listagem dos scripts de inicialização é encontrada em Appendix D.

- Seção 7.6, “LFS-Bootscripts-20130821.”
- Seção 7.7, “Como esses scripts de inicialização funcionam?.”
- Seção 7.8, “Configurando o hostname do sistema.”
- Seção 7.9, “Configurando o Script Setclock.”
- Seção 7.10, “Configurando o Terminal Linux.”
- Seção 7.11, “Configurando o script sysklogd.”

Finalmente, há uma breve introdução aos scripts e arquivos de configuração usados quando o usuário efetua logon no sistema.

- Seção 7.13, “Os Arquivos de inicialização do Bash Shell.”
- Seção 7.14, “Criando o Arquivo /etc/inputrc.”

7.2. Configuração Geral de Rede

Esta seção somente aplica-se a quem irá configurar uma placa de rede.

Se uma placa de rede não será utilizada, não há necessidade de criar nenhum arquivo de configuração relacionado a placas de rede. Se este for o caso, você precisará remover os symlinks `network` de todos os diretórios de níveis de execução (`/etc/rc.d/rc*.d`) após os scripts de inicialização estarem instalados em Seção 7.6, “LFS-Bootscripts-20130821”.

7.2.1. Criando nomes estáveis para interfaces de rede

Se houver apenas uma interface de rede a ser configurada, esta seção é opcional, ainda que nunca será errado seguila. Em muitos casos (e.g. um laptop com uma interface sem fio e outra com fio), executar a configuração nesta seção é necessário.

Com o Udev e drivers de rede modulares, a numeração da interface de rede não é persistente por padrão através das reinicializações, devido ao fato dos drivers serem carregados em paralelo e, portanto, em ordem randômica. Por exemplo, em um computador contendo duas placas de rede feitas pela Intel e Realtek, a placa de rede manufaturada pela inel pode se tornar `eth0` e a placa da Realtek se torna `eth1`. Em alguns casos, após uma reinicialização a placa é renumerada trocando os valores um pelo outro. Para evitar isso, o Udev vem com um script e algumas regras para atribuir nomes estáveis para placas de rede baseado em seu endereço MAC.

As regras foram geradas previamente nas instruções de compilação para udev (systemd) no último capítulo. Inspecione o arquivo `/etc/udev/rules.d/70-persistent-net.rules`, para descobrir que nome foi atribuído para cada dispositivo de rede:

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

Nota

Em alguns casos como quando o endereço MAC foi atribuído a uma placa de rede manualmente ou em um ambiente virtual como Xen, os arquivos de regras de rede podem não ter sido gerados porque os endereços não está consistentemente atribuídos. Neste caso, apenas continue até a próxima seção.

O arquivo começa com um bloco de comentário seguido por duas linhas para cada NIC. A primeira linha para cada NIC é uma descrição comentada mostrado seus IDs de hardware (e.g. seu PCI vendor e IDs de dispositivos, se isso for uma placa PCI), juntamente com seu driver em parenteses, se o driver pode ser encontrado. Nem o ID do hardware nem o driver são usados para determinar que nome deve ser dado a uma interface; esta informação é apenas para referência. A segunda linha é a regra Udev que bate com esse NIC e atribui um nome a ele.

Todas as regras Udev são compostas com várias chaves, separadas por vírgulas e espaços em branco opcionais. As chaves de regras e uma explicação sobre cada uma delas estão a seguir:

- `SUBSYSTEM=="net"` - Isso diz ao Udev para ignorar dispositivos que não são placas de rede.
- `ACTION=="add"` - Isso diz ao Udev para ignorar esta regra para uevent que não é uma adição (uevents "remove" e "change" também acontecem, mas não precisam renomear interfaces de rede).
- `DRIVERS=="?*"` - Isso existe para que Udev ignore VLAN ou sub-interfaces bridge (porque essas sub-interfaces não têm drivers). Essas sub-interfaces são puladas porque o nome que seria atribuído a elas colidiria com seus dispositivos pais.
- `ATTR{address}` - O valor desta chave é o endereço MAC do NIC.
- `ATTR{type}=="1"` - Isso garante que a regra corresponda apenas a interface primária no casa de certos drivers wireless, que criam múltiplas interfaces virtuais. As interfaces secundárias são puladas pela mesma razão que VLAN e sub-interfaces bridge são puladas: haveria uma colisão de noves de outra forma.

- KERNEL=="eth*" - Isso foi adicionado ao gerador de regras Udev para manipular máquinas que tem múltiplas interfaces de rede, todas com o mesmo endereço MAC (o PS3 é uma máquina dessas). Se a interface independente tem diferentes basenames, esta chave permitirá ao Udev distinguí-las. Isso não é necessário para a maioria dos usuários do Linux From Scratch, mas não machuca.
- NAME - O valor desta chave é o nome que o Udev irá atribuir a essa interface.

O valor de NAME é a parte importante. Certifique-se de saber que nome foi atribuído a cada uma das suas placas de rede antes de prosseguir, e certifique-se de usar o valor NAME quando criando seus arquivos de configuração abaixo.

7.2.2. Criando Arquivos de Configuração de Interface de Rede

Que interfaces são levantadas ou derrubadas pelo script de rede depende dos arquivos em /etc/sysconfig/. Esse diretório deve conter um arquivo para cada interface a ser configurada, tal como ifconfig.xyz, onde “xyz” é significante para o administrador assim como o nome do dispositivo (e.g. eth0). Dentro desse arquivo estão atributos para esta interface, tais como seu endereço(s) IP, máscara de subrede, e por aí vai. IÉ necessário que a base no nome do arquivo seja *ifconfig*.

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

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

Os valores dessas variáveis deve ser mudado em cada arquivo para bater com a configuração adequada.

Se a variável ONBOOT está configurada como “yes” o script de rede levantada o "Network Interface Card" (NIC) durante a inicialização do sistema. Se configurado para qualquer coisa exceto “yes” o NIC será ignorado pelo script de rede e não será levantado automaticamente. A interface pode ser manualmente iniciada ou parada com os comandos **ifup** e **ifdown**.

A variável IFACE define o nome da interface, por exemplo, eth0. É necessário para todos os arquivos de configuração de dispositivos de rede.

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

A variável GATEWAY deve conter o valor padrão do IP do gateway, se um estiver presente. Se não, então comente a variável totalmente.

A variável PREFIX contém o número de bits usados na subrede. Cada octeto usado em um endereço IP tem 8 bits. Se a máscara de 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, ela estaria usando os primeiros 28 bits. Prefixos mais longos que 24 são comumente usados por DSL e Internet Service Providers (ISPs) baseados em cabos. neste exemplo (PREFIX=24), a máscara de rede é 255.255.255.0. Ajuste a variável PREFIX de acordo com seu subnet específico. Se omitido, o PREFIX padrão é 24.

Para mais informações veja a página de manual **ifup**.

7.2.3. Criando o Arquivo /etc/resolv.conf

Se o sistema for conectado a internet, ele precisará de alguma forma de resolução de DNSs para resolver nomes de domínios de internet para endereços IP, e vice versa. Isso é melhor alcançado colocando endereços de IP de servidores de DNS, disponíveis do ISP ou do administrador, em `/etc/resolv.conf`. Crie o arquivo rodando 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 a página de manual para resolv.conf para mais detalhes.

Substitua *<IP address of the nameserver>* com o endereço IP para o DNS mais apropriado para a configuração. Haverá frequentemente mais de uma entrada (requisitos exigem servidores secundários para capacidade em caso de queda). Se precisa ou quer apenas um servidor de DNS, remova a segunda linha *nameserver* do arquivo. O endereço IP pode também ser um router em uma rede local.

Nota

Os endereços de DNS IPv4 públicos do Google são 8.8.8.8 e 8.8.4.4.

7.3. Customizando o Arquivo /etc/hosts

Se uma placa de rede será configurada, decida quanto ao endereço IP, o FQDN, e possível apelidos (aliases) a serem utilizados e que devem ser colocados no arquivo `/etc/hosts`. A sintaxe é:

```
IP_address myhost.example.org aliases
```

A menos que o computador deva estar visível na internet (i.e., há um domínio registrado e um bloco válido de endereço IP atribuído—a maioria dos usuários não têm isso), certifique-se que o endereço IP esteja dentro da faixa de IP privado da rede. Faixas válidas 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 na faixa 16-31. y pode ser qualquer número na faixa 0-255.

Um endereço de IP privado válido poderia ser 192.168.1.1. Um FQDN válido para este ip poderia ser lfs.example.org.

Ainda que não se use uma placa de rede, um FQDN válido ainda é necessário. Isso é necessário para que certos programas operem corretamente.

Crie o arquivo `/etc/hosts` executando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# End /etc/hosts (network card version)
EOF
```

Os valores *<192.168.1.1>* e *<HOSTNAME.example.org>* precisam ser mudados para usos específicos ou necessidades (se atribuído um endereço de IP pela rede/administrador do sistema e a máquinas será conectada a uma rede existente). Os apelidos opcionais podem ser omitidos .

Se uma placa de rede não será configurada, crie o arquivo `/etc/hosts` rodando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

7.4. Manipulando dispositivos e módulos em um Sistema LFS

No Capítulo 6, nós instalamos o pacote Udev. Antes de entrarmos nos detalhes a respeito de como ele trabalha, vamos fazer um breve histórico dos métodos de manipulação de dispositivos.

Os sistemas Linux geralmente usam o tradicional método de criação estática de dispositivos, pelo qual muitos nós de dispositivos (device nodes) são criados no diretório `/dev` (às vezes milhares de nós, literalmente), independente de o dispositivo de hardware correspondente existir ou não. Isto é tipicamente feito pelo script **MAKEDEV**, que faz algumas chamadas ao programa **mknod** com a identificação numérica principal e secundária dos dispositivos para cada dispositivo que possa existir no mundo.

Usando o método Udev, somente aqueles dispositivos que são destacados pelo kernel inicializam os nós de dispositivo criados para eles. Como estes nós de dispositivos são criados a cada inicialização do sistema, serão armazenados em um sistema de arquivos do tipo `devtmpfs` (um sistema de arquivos virtual que reside inteiramente na memória do sistema). Os nós do dispositivo não exigem muito espaço, assim a memória usada é insignificante.

7.4.1. Histórico

Em fevereiro 2000, um novo sistema de arquivos chamado `devfs` foi incorporado no kernel 2.3.46 e tornado disponível na séries 2.4 de kernels estáveis. Embora estivesse presente no próprio código-fonte do kernel, este método de criar dispositivos dinamicamente nunca recebeu o suporte decisivo dos desenvolvedores do núcleo do kernel.

O problema principal com a abordagem adotada pelo devfs era o modo como manipulava a detecção, criação e nomeação de dispositivos. Esta última etapa, quando o nó de dispositivo recebe um nome, era talvez a mais crítica. Como os nomes dos dispositivos são passíveis de configuração, seria aceitável então que a política para dar nomes aos dispositivo fosse do administrador do sistema e não imposta por algum desenvolvedor qualquer. O sistema de arquivos devfs sofre também com algumas condições que são inerentes ao seu projeto e não podem ser eliminadas sem uma revisão substancial do kernel. Ficou marcado também pela desatualização – devido à falta de manutenção – e foi finalmente removido do kernel em junho de 2006.

Com o desenvolvimento das versões instáveis 2.5 do kernel, liberado mais tarde com a série estável 2.6 do kernel, surge um novo sistema de arquivos virtual chamado sysfs. O trabalho do sysfs é fornecer uma visão da configuração do hardrware do sistema para os processos, ao nível do usuário. Com esta nova representação (userspace-visible representation), a possibilidade de ser vista uma modificação no devfs ao nível do usuário tornou-se muito mais realista.

7.4.2. Implementação Udev

7.4.2.1. Sysfs

O sistema de arquivos sysfs foi rapidamente mencionado anteriormente. Mas como o sysfs pode saber sobre os dispositivos existentes no sistema e que identificações numéricas de dispositivos usar. Os drivers que tenham sido compilados no kernel registram diretamente seus objetos com um sysfs (devtmpfs internamente) assim que são detectados pelo kernel. Para os drivers compilados como módulos, este registro acontecerá quando o módulo é carregado. Assim que o sistema de arquivos sysfs é montado (em /sys), os dados que os drivers internos registraram com sysfs estão disponíveis para os processos ao nível de usuário e para o processamento do udev (incluindo modificações para nós de dispositivos).

7.4.2.2. Criação de Nós de Dispositivos

Arquivos de dispositivos são criados pelo sistema de arquivos devtmpfs. Qualquer driver que deseje registrar um nó de dispositivo usará devtmpfs(via driver core) para fazê-lo. Quando uma instância de devtmpfs é montada em /dev, o nó de dispositivo será inicialmente criado com um nome fixo, permissões e dono.

Pouco tempo depois, o kernel enviará um uevent para **udevd**. Baseado nas regras especificadas nos arquivos dentro dos diretórios /etc/udev/rules.d, /lib/udev/rules.d, e /run/udev/rules.d, **udevd** criará symlinks adicionais para os nós de dispositivos, ou mudará suas permissões, dono, ou grupo, ou modificará a entrada no banco de dados interno **udevd** (nome) para aquele objeto.

Essas regras nesses três diretórios são numeradas de modo similar ao pacote LFS-Bootscripts e todos os três diretórios são fundidos. Se **udevd** não puder encontrar uma regra para o dispositivo que ele está criando, ele deixará as permissões e posse no valor que devtmpfs usou inicialmente.

7.4.2.3. Bootscripts Udev

primeiro bootscript do LFS, /etc/init.d/mountvirtfs irá copiar quaisquer nós de dispositivos em /lib/udev/devices para /dev. Isso é necessário porque alguns dispositivos, diretórios, e symlinks são necessários antes que os processos que manuseiam dispositivo dinamicamente estejam disponíveis durante os estágios iniciais do sistema em inicialização, ou sejam necessários pelo **udevd**. Criar nós de dispositivos estáticos em /lib/udev/devices também provê uma maneira fácil de lidar com dispositivos não suportados pela infraestrutura dinâmica de manuseio de dispositivos.

O initiscript `/etc/rc.d/init.d/udev` inicia o **udevd**, ativando qualquer dispositivo "coldplug" que já tenha sido criado pelo kernel e espera que qualquer regra seja completada. Este script também modifica o manipulador de eventos do udev alterando o padrão `/sbin/hotplug` . Isto é feito porque o kernel não precisa mais chamar uma biblioteca externa. Em vez disso **udevd** irá escutar em um socket netlink os uevents que o kernel emite.

O initiscript **/etc/rc.d/init.d/udev_retry** icuida dos eventos de reativação para subsistemas cujas regras podem depender de sistemas de arquivos que não estão montados até que o script **mountfs** seja executado (em particular, `/usr` e `/var` podem causar isso). Este script roda depois do script **mountfs** então aqueles regras (se acionadas novamente) devem funcionar nesta segunda vez. Se ele estiver configurado a partido do arquivo `/etc/sysconfig/udev_retry`; quaisquer palavras neste arquivo que não sejam comentários são consideradas nomes de subsistemas a serem ativados no momento da segunda tentativa. Para encontrar o subsistema de um dispositivo, use **udevadm info --attribute-walk <device>** onde <device> é um caminho absoluto em /dev ou /sys tal como /dev/sr0 ou /sys/class/rtc.

7.4.2.4. Carregamento de Módulos

Drivers de dispositivos compilados como módulos podem ter apelidos (alias) construídos entro deles. Apelidos (alias) são visíveis na saída do programa **modinfo** e são geralmente relacionadas aos identificadores (bus-specific)dos dispositivos suportados pelo módulo. Por exemplo, o driver *snd-fm801* suporta dispositivos PCI com vendor ID 0x1319 e ID de dispositivo 0x0801, e tem um alias de “pci:v00001319d00000801sv*sd*bc04sc01i*”. Para a maioria dos dispositivos, o driver exporta a alias do driver que manipularia o dispositivo via `sysfs`. E.g., o arquivo `/sys/bus/pci/devices/0000:00:0d.0/modalias` pode conter a string “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. As regras disponíveis por padrão no Udev farão **udevd** chamar **/sbin/modprobe** com o conteúdo das variáveis de ambiente uevent do `MODALIAS` uevent environment variable (que devem ser as mesmas do arquivo `modalias` em sysfs), dessa forma carregando todos os módulos cujas alias correspondem depois da expansão do wildcard.

Neste exemplo, isso mesmo, em adição a *snd-fm801*, o obsoleto (e indesejado) driver *forte* será carregado se estiver disponível. Veja abaixo maneiras de prevenir o carregamento de drivers indesejados.

O kernel por si só é capaz de carregar módulos para suportes a protocolos de rede, sistemas de arquivo e NLS sob demanda.

7.4.2.5. Manuseando Dispositivos Dinâmicos (Hotpluggable/Dynamic)

Quando você conecta um dispositivo, como um MP3 Player USB, o kernel reconhece que o dispositivo está conectato e gera um uevent. Esse uevent é então tratado por **udevd** como descrito acima.

7.4.3. Problemas ao Carregar Módulos e Criar Dispositivos

Há alguns possíveis problemas quando se trata de criar automaticamente nós de dispositivos.

7.4.3.1. Um módulo do kernel não é carregado automaticamente

O Udev só carregará um módulo se ele tiver uma alias específica para barramento (bus-specific) e o driver de barramento exportar devidamente as alias para `sysfs`. Em outros casos, deve-se conseguir o carregamento do módulo por outros meios. Com o Linux-3.13.3, Udev é conhecido por carregar driver escritos adequadamente para dispositivos INPUT, IDE, PCI, USB, SCSI, SERIO, e FireWire.

Para determinar se o dispositivos que você requer tem o suporte necessário para Udev, execute **modinfo** com o nome do módulo como argumento. Agora tente localizar o dispositivo em `/sys/bus` e verifique se há um arquivo `modalias` lá.

Se o arquivo `modalias` existe em `sysfs`, o driver suporta o dispositivo e pode falar com ele diretamento, mas não tem a alias, o que é um bug no driver. Carregue o driver sem a ajuda do Udev e espere que o problema seja resolvido posteriormente.

Se não houver nenhum arquivo `modalias` no diretório relevante `/sys/bus`, isso significa que os desenvolvedores do kernel ainda não adicionaram modalias suporte para este tipo de barramento. Com Linux-3.13.3, esse é o caso com barramentos ISA. Espere que esse problema seja resolvido em versões posteriores do kernel.

Udev não se destina a carregar drivers “wrapper” tais como *snd-pcm-oss* e drivers non-hardware tais como *loop* de maneira alguma.

7.4.3.2. Um módulo do kernel não é carregado automaticamente, e Udev não se destina a carregado

Se o módulo “wrapper” apenas aumenta a funcionalidade provida por outro módulo (e.g., *snd-pcm-oss* aumenta a funcionalidade de *snd-pcm*azendo placas de som disponíveis para aplicações OSS), configure **modprobe** para carregar um encapsulador após o Udev carregar o módulo encapsulado. Para fazer isso, adicione uma linha “softdep” em qualquer arquivo `/etc/modprobe.d/<filename>.conf`. Por exemplo:

```
softdep snd-pcm post: snd-pcm-oss
```

Note que o comando “softdep” também permite dependências `pre:` ou uma mistura de ambos `pre:` e `post:`. Veja a página de manual `modprobe.d(5)` para mais informações sobre a sintaxe e funcionalidades “softdep”.

Se o módulo em questão não é um encapsulador e é usado por ele mesmo, configure o script de inicialização **modules** para carregar esse módulo na inicialização do sistema. Para fazer isso, adicione o nome do módulo ao arquivo `/etc/sysconfig/modules`em uma linha separada. Isso funciona para módulos encapsuladores também, mas é sub-ótima naquele caso.

7.4.3.3. Udev carrega alguns módulos indesejados

Ou não construa o módulo, ou coloque-o em uma blacklist em um arquivo `/etc/modprobe.d/blacklist.conf` como feito com o módulo *forte* no exemplo abaixo:

```
blacklist forte
```

Módulos em blacklists ainda podem ser carregados manualmente com o comando explícito **modprobe**.

7.4.3.4. Udev cria um dispositivo incorretamento, ou faz um symlink errado

Isso geralmente acontece se uma regra inesperada confere com o dispositivo. Por exemplo, uma regra mal escrita pode valer tanto para um disco SCSI (como desejado) e para um dispositivo SCSI genérico (incorreto) pelo vendedor (vendor). Encontre a regra ofensiva e faça-a mais específica, com a ajuda do comando **udevadm info**.

7.4.3.5. Regras Udev funcionam de modo não confiável

Isso pode ser outra manifestação do problema anterior. Caso não, e sua regra usa atributos `sysfs`, isso pode ser um problema no kernel, a ser consertado em próximos kernels. Por hora, você pode resolver criando regras que esperam pelos atributos `sysfs` usados e adicionando-os 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.

7.4.3.6. Udev Não cria um dispositivo

O texto que se segue assume que o driver é construído estaticamente denstro ddo kernel ou já está carregado como um módulo, e que você já verificou que o Udev não cria um dispositivo nomeado erroneamente.

Udev não tem qualquer informação necessária para criar um nó de dispositivo se um driver do kernel não exporta seus dados para `sysfs`. Isso é mais comum com drivers de terceiros de fora da árvore do kernel. Crie um nó de dispositivo estático em `/lib/udev/devices` om os números major/minor apropriados (veja o arquivo `devices.txt` dentro da documentação do kernel ou a documentação disponibilizada pelo distribuidor do driver). O nó de dispositivo estático será copiado para `/dev` pelo script de inicialização **udev**.

7.4.3.7. Ordem de nomeação de dispositivos muda aleatoriamente após reinicializar

Isso se deve ao fato de o Udev, por design, manipular uevents e carregar módulos em paralelo, e dessa forma em uma ordem imprevisível. Isso nunca será "consertado". Você não deve confiar que os nomes de dispositivos do kernel sejam estáveis. Em vez disso, crie suas próprias regras que criam symlinks com nomes estáveis baseados em alguns atributos estáveis do dispositivo, tais como um número de série ou a saída de vários *_id instalados pelo Udev. Veja Seção 7.5, "Criando Symlinks para Dispositivos" e Seção 7.2, "Configuração Geral de Rede" para exemplos.

7.4.4. Leitura Útil

Documentação de ajuda adicional está disponível nos seguintes sites:

- 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*

7.5. Criando Symlinks para Dispositivos

7.5.1. Symlinks de CD-ROM

Alguns programas que você pode querer instalar posteriormente (e.g., vários reprodutores de mídia) esperam que os symlinks `/dev/cdrom` e `/dev/dvd` existam, e apontem para um dispositivo de CD-ROM ou DVD-ROM. Também pode ser conveniente colocar referências a esses symlinks em `/etc/fstab`. Udev vem com um script que irá gerar arquivos de regras que criam esses symlinks para você, dependendo das capacidades de cada dispositivo, mas você precisa decidir qual dos dois modos de operação você deseja que o script use.

Primeiro, o script pode operar no modo "by-path" (usado por padrão por 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 "by-id" (padrão para dispositivos IDE e SCSI), onde as regras que são criadas dependem das strings de identificação armazenadas no próprio dispositivo de CD ou DVD. O path é determinado pelo script **path_id** do Udev, e as strings de identificação são lidas do hardware pelos programas **ata_id** ou **scsi_id**, dependendo de que tipo de dispositivo você tenha.

Há vantagens em cada abordagem; a abordagem correta a ser utilizada depende de que tipos de mudanças de dispositivos acontecerão. Se você espera o caminho físico para o dispositivo (isto é, as portas e/ou slots aos quais será conectado) mudem, por exemplo porque você planeja mudar o driver para uma porta IDE diferente ou para um conector USB diferente, então você deveria usar o modo "by-id". Em contra partida, se você espera que a identificação do dispositivo mude, por exemplo porque ele pode morrer, e você o substituiria por um dispositivo diferente com as mesmas capacidades e que seria plugado no mesmo conector, então você deveria usar o modo "by-path".

Se ambos os tipos de mudanças são possíveis com seu driver, então escolha um modo baseado no tipo de mudança que você espera que aconteça com maior freqüência.

Importante

Dispositivos externos (por exemplo, um drive de CD conectado através de USB) não deve usar persistência por caminho, porque cada vez que o dispositivo for plugado em uma nova porta externa, seu caminho físico irá mudar. Todos os dispositivos conectados externamente terão esse problema se você escrever as regras Udev para reconhecê-los pelo caminho físico; o problema não se limita a drives de CD e DVD.

Se você quiser ver os valores que os scripts Udev irão utilizar, então para o dispositivo de CD-ROM apropriado, encontre o diretório sob `/sys` (e.g., pode ser `/sys/block/hdd`) e rode um comando similar a:

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

Veja as linhas contendo a saída de vários programas *_id. O modo “by-id” usará o valor ID_SERIAL se ele existir e não estiver vazio, de outro modo ele irá usar uma combinação de ID_MODEL e ID_REVISION. O modo “by-path” usará o valor ID_PATH.

Se o valor padrão não for adequado para a situação, então a seguinte modificação pode ser feita para o arquivo `/etc/udev/rules.d/83-cdrom-symlinks.rules`, acomo se segue (onde *`mode`* é um dos “by-id” ou “by-path”):

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

Note que não é necessário criar os arquivos de regras para os symlinks neste momento, porque você montou com bind o diretório do sistema anfitrião `/dev` dentro do sistema LFS, e nós assumimos que esses symlinks existem no anfitrião. As regras e os symlinks serão criados na primeira vez que você inicializar seu sistema LFS.

Entretanto, se você tiver múltiplos drives de CD-ROM, então os symlinks gerados naquele momento podem aponto para dispositivos diferentes daqueles que eles apontavam em seu sistema anfitrião, porque os dispositivos não são descobertos em uma ordem previsível. Essas atribuições criadas quando você inicializa o sistema LFS pela primeira vez serão estáveis, então isto é um problema apenas se você precisar dos symlinks em ambos os sistemas para apontar para o mesmo dispositivo. Se você precisa disso, então inspecione (e possivelmente edite) o arquivo `/etc/udev/rules.d/70-persistent-cd.rules` gerado após a inicialização, para ter certeza que as atribuições de symlinks correspondem ao que você precisa.

7.5.2. Lidando com dispositivos duplicados

Como explicado em Seção 7.4, “Manipulando dispositivos e módulos em um Sistema LFS”, a ordem em que dispositivos com a mesma função aparecem em `/dev` é essencialmente randômica. E.g., se você tem uma câmera web USB e um controle de TV, às vezes `/dev/video0` refere-se à câmera e `/dev/video1` refere-se ao controle, e às vezes após a reinicialização a ordem muda para o inverso. Para todas as classes de hardware exceto placas de com e de rede, isso pode ser ajustado criando regras udev para symlinks persistentes. O caso de placas de rede é abordado separadamente em Seção 7.2, “Configuração Geral de Rede”, e configuração de placas de som pode ser encontrado em *BLFS*.

Para cada um dos seus dispositivos que pode ter esse problema (mesmo que o problema não exista em sua distribuição Linux atual), encontre o diretório correspondente em `/sys/class` ou `/sys/block`. Para dispositivos de vídeo, ele pode ser `/sys/class/video4linux/videoX`. Descubra quais os atributos que identificam o dispositivo de maneira única (geralmente, vendor e IDs de produtos e/ou números seriais funcionam):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Então escreva regras que criam os symlinks, e.g.:

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Persistent symlinks for webcam and tuner
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f", ATTRS{vendor}=="0x109e", \
    SYMLINK+="tvtuner"

EOF
```

O resultado é que os dispositivos `/dev/video0` e `/dev/video1` ainda referem-se aleatoriamente para o controle a câmera (e portanto nunca devem ser usados diretamente), mas há symlinks `/dev/tvtuner` e `/dev/webcam` que sempre apontam para o dispositivo correto.

7.6. LFS-Bootscripts-20130821

O pacote LFS-Bootscripts contém um conjunto de scripts de inicialização/finalização do sistema LFS durante o bootup/shutdown.

Tempo de Construção: menos de 0.1 SBU
Espaço de disco: 260 KB

7.6.1. Instalação do LFS-Bootscripts

Instale o pacote:

```
make install
```

7.6.2. 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 instalado: /etc/rc.d, /etc/init.d (symbolic link), /etc/sysconfig, /lib/services, /lib/lsb (symbolic link)

Breve descrição

checkfs Verifica a integridade dos sistemas de arquivos antes que sejam montados (exceto os sistemas de arquivos baseados em journal e rede).

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

console Carrega a tabela correta do keymap para o layout de teclado desejado; ajusta também a fonte de tela

functions Contém as funções comuns, como as de verificação de erro e status, que são usadas por diversos scripts

halt Desliga o sistema

ifdown Para um dispositivo de rede

ifup Inicializa um dispositivo de rede

localnet Define os dispositivos hostname e local loopback do sistema

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

mountfs Monta todos os sistemas de arquivos, exceto os que estão definidos como *noauto* e os baseados em rede

mountvirtfs Monta os sistemas de arquivo virtuais do kernel, como o `proc`

network Configura as interfaces de rede, como a placa de rede, e define o gateway padrão (onde aplicável).

rc O script mestre de controle dos níveis de execução (run-level); é responsável pela execução de todos os demais bootscripts, um a um, em uma seqüência determinada pelo nome das ligações simbólicas que estão sendo processadas

reboot Reinicializa o sistema

sendsignals Certifica-se que cada processo está terminado antes que o sistema reinicialize ou desligue

setclock Ajusta o relógio do kernel para a hora local quando o relógio do hardware não está ajustado com a hora UTC

ipv4-static Fornece a funcionalidade necessária para atribuir um endereço IP (Internet Protocol) estático para uma interface de rede

swap Habilita e desabilita os arquivos e a partição de troca (swap)

sysctl Carrega valores de configuração de sistema do `/etc/sysctl.conf`, se esse arquivo existir, dentro do kernel em execução

sysklogd Inicia e finaliza os log daemons do sistema e do kernel

template Um modelo para criar scripts de inicialização padronizados para outros serviços do sistema

udev Prepara o diretório `/dev` e inicializa o Udev

udev_retry Tenta novamente uevents do udev que falharam, e copia os arquivos de regras gerados para `/etc/udev/rules.d` se necessário

7.7. Como esses scripts de inicialização funcionam?

Linux usa um aparato de inicialização especial nomeado SysVinit que é baseado em um conceito de *run-levels*. Isso pode ser bem diferente de um sistema para outro, então não se pode assumir que porque as coisas funcionam em uma distribuição Linux em particular elas deveriam funcionar da mesma forma no LFS. LFS tem sua própria maneira de fazer as coisas, mas ele respeita os padrões aceitos em geral.

SysVinit (ao qual pode-se referir como “init” daqui pra frente) funciona usando um esquema de níveis de execução (run-levels). Existem sete (numerados de 0 a 6) run-levels (na realidade existem mais run-levels, mas eles são para casos especiais e não são usados geralmente. Veja `init(8)` para mais detalhes), e cada um deles corresponde as ações que o computador deve executar quando é ligado. O run-level padrão é 3. Aqui estão as descrições dos diferentes run-levels conforme eles são implementados:

0: desligar o computador
1: modalidade mono-usuário
2: modalidade multi-usuário sem suporte à rede
3: modalidade multi-usuário com suporte à rede
4: reservado para configuração, mas faz o mesmo que 3
5: mesmo que 4, é usado geralmente para o início de uma sessão do GUI (como o **xdm** do X ou **kdm** do KDE)
6: reinicia o computador

7.7.1. Configurando o Sysvinit

Durante a inicialização do kernel, o primeiro programa que é executado é também especificado na linha de comando ou, por padrão **init**. Esse programa 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 para esse 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 irá instruir **rc** a rodar todos os scripts começando com S no diretório `/etc/rc.d/rcS.d` seguido por todos os scripts começando com S no diretório `/etc/rc.d/rc?.d` onde a marca de interrogação é especificada pelo valor initidefault.

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`. Quaquer dos parâmetros de configuração dos arquivos do sistema descritos em seções subsequentes pode ser colocado alternativamente neste arquivo permitindo consolidação de todos os parâmetros do sistema neste único arquivo.

Como uma conveniência para depuração, os scripts de funções também registram todas as saídas para `/run/var/bootlog`. Uma vez que o diretório `/run` é um tmpfs, esse arquivo não é persistente através de inicializações, entretanto ele é adicionado ao arquivo mais permanente `/var/log/boot.log` ao final do processo de inicialização.

7.7.2. Mudando Run levels

Mudança de run-levels é feita com **init <*runlevel*>**, onde <*runlevel*> é o run-level desejado. Por exemplo, para reiniciar o computador, um usuário poderia utilizar o comando **init 6**, o qual é uma alias para o comando **reboot**. Da mesma forma, **init 0** é uma alias para o comando **halt**.

Há vários diretórios em `/etc/rc.d` que parecem com `rc?.d` (onde ? é o número do runlevel) e `rcsysinit.d`, todos contendo uma certa quantidade 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 (start) um serviço. Os números determinam a ordem na qual os scripts são executados, de 00 a 99—quanto menos o número mais cedo ele é executado. Quando **init** muda para outro run-level, os serviços adequados são tanto iniciados ou parados, dependendo do runlevel escolhido.

Os scripts reais estão em `/etc/rc.d/init.d`. Eles fazem o trabalho real, e todos os symlinks apontam para eles. K links e S links apontam para o mesmo script em `/etc/rc.d/init.d`. Isso acontece porque os scripts podem ser chamados com diferentes parâmetros 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`* argument.

Há uma excessão para essa explicação. Links que começam com um *S* nos diretórios `rc0.d` e `rc6.d` não farão nada iniciar. Eles serão chamados com o parâmetro *`stop`* para parar alguma coisa. A lógica atrás disso é que quando um usuário está para reiniciar ou desligar o sistema, nada precisa ser iniciado. O sistema apenas precisa ser parado.

Essas são descrições do que os argumentos provocam nos scripts:

`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 atuaizada. Isso é usado após a modificação de um arquivo de configuração de serviço, quando o serviço não precisa ser reiniciado.

`status`
: Diz se o serviço está rodando e qual seus PIDs.

Sinta-se livre para modificar a maneira como o processo de inicialização funciona (aginal de contas, este é seu próprio sistema LFS). Os arquivos dados aqui são um exemplo de como isso pode ser feito.

7.8. Configurando o hostname do sistema

Parte do trabalho do script **localnet** é configurar o nome do computador (hostname). Isso precisa ser configurado no arquivo `/etc/sysconfig/network`.

Crie o arquivo `/etc/sysconfig/network` e entre com um hostname executando:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

<*lfs*> precisa ser substituído com o nome dado para o computador. Não insira o Fully Qualified Domain Name (FQDN) aqui. Esta informação é colocada no arquivo `/etc/hosts` file.

7.9. Configurando o Script Setclock

O script **setclock** obtém a data/hora do relógio do computador, também conhecido como relógio do BIOS ou Complementary Metal Oxide Semiconductor (CMOS). Se o relógio do computador estiver ajustado ao UTC, este script converterá a data/hora para o horário local usando o arquivo `/etc/localtime` (que diz ao programa **hwclock** qual a timezone do usuário). Não há nenhuma maneira de se detectar se relógio do computador está ou não o ajustado ao UTC, isto precisa ser configurado manualmente.

O comando **setclock** é executado via udev quando o kernel detecta a capacidade do hardware durante o boot. Também pode ser rodado manualmente com o parâmetro stop para armazenar a hora do sistema para o relógio CMOS.

Se você não sabe se o relógio do computador está ajustado ao UTC, descubra com o comando **`hwclock --localtime --show`**. Isto mostrará a data/hora de acordo com seu compudador. Se estiver de acordo com o seu relógio, o relógio do seu sistema está ajustado à hora local. Se a saída do **hwclock** não for a hora local, é possível que seu computador esteja ajustado ao UTC. Verifique isto adicionando ou subtraindo a quantidade apropriada de horas conforme a timezone mostrada pelo **hwclock**. Por o exemplo, se você estiver atualmente no timezone MST, conhecido também como GMT -0700, adicione sete horas a sua hora local.

Mude o valor da variável `UTC` abaixo para *`0`* (zero) se o relógio do sistema *não* estiver ajustado ao 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 o tempo no LFS está disponível em *http://www.linuxfromscratch.org/hints/downloads/files/time.txt*. Isso explica questões como timezones (fusos horários), UTC, e a variável de ambiente `TZ`.

Nota

Os parâmetros CLOCKPARAMS e UTC podem ser configurados alternativamente no arquivo `/etc/sysconfig/rc.site`.

7.10. Configurando o Terminal Linux

Esta seção discute como configurar o script de inicialização **console** que define o mapa do teclado, a fonte do terminal e o nível de registro do console do kernel. Se caracteres não-ASCII (por exemplo, o sinal britânico da libra e o símbolo monetário do euro) não forem usados, e o teclado for do padrão U.S. , salte esta seção. Sem um 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 obter as informações de configuração. Decida qual a fonte de tela e o keymap que devem ser utilizados. Vários HOWTOs específicos para certas línguas podem ajudar nesta tarefa, veja *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Se ainda tiver dúvidas, procure nos diretórios `/usr/share/keymaps` e `/usr/share/consolefonts` por keymaps válidos e fontes. Read `loadkeys(1)` and `setfont(8)` manual pages to determine the correct arguments for these programs.

O arquivo `/etc/sysconfig/console` deve conter linhas na forma: VARIABLE="valor". As seguintes variáveis são reconhecidas:

LOGLEVEL
: Esta variável especifica o log level para mensagens do kernel enviadas para o terminal como configurado por **dmesg**. Níveis válidos vão de "1" (nenhuma mensagem) até "8". O nível padrão é "7".

KEYMAP
: Esta variável especifica os parâmetros para o programa **loadkeys**, tipicamente, o nome do keymap a carregar, e.g., “es”. Se esta variável não estiver configurada, o bootscriot não vai rodar o programa **loadkeys**, e o keymap padrão do kernel será usado.

KEYMAP_CORRECTIONS
: Essa variável (raramente usada) especifica os parâmetros para a segunda chamada ao programa **loadkeys**. Isso é útil se o keymap padrão não for completamente satisfatório e um pequeno ajuste tenha que ser feito. E.g., para incluir o símbolo Euro em um keymap que normalmente não o possui, configure esta variável para “euro2”.

FONT
: Essa variável especifica os parâmetros para o programa **setfont**. Tipicamente, isso inclui o nome da fonte, “-m”, e o nome do mapa de caracteres a ser carregado. E.g., para carregar a fonte “lat1-16” juntamente com o mapa de caracteres “8859-1” uma vez que ele é apropriado nos EUA), configure essa variável para “lat1-16 -m 8859-1”. Em modo UTF-8, o kernel usa o mapa de caracteres para conversão de códigos de 8-bits compostos para UTF-8, e assim o argumento do parâmetro "-m" deveria ser configurado para a codificação dos códigos compostos no keymap.

UNICODE
: Ajuste esta variável para “1”, “yes” ou “true” para colocar o terminal em modo UTF-8. Isso é útil em locales baseados em UTF-8 e inofensivo de outra forma.

LEGACY_CHARSET
: Para muitos layouts de teclado, não há um keymap Unicode regular no pacote Kbd. O script de inicialização **console** irá converter um keumap disponível para UTF-8 seesta variável estiver configurada para a codificação não-UTF-8 disponível do keymap.

Alguns exemplos:

- Para uma configuração não-Unicode, apenas as variáveis KEYMAP e FONT são geralmente necessárias. E.g., 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, às vezes é necessário ajustar um keymap. O exemplo seguinte adiciona o símbolo Euro ao keymap alemão:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# End /etc/sysconfig/console
EOF
```

- O seguinte é um exemplo habilitado para Unicode para búlgaro, onde um keymap regular UTF-8 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 da fonte 512-glyph LatArCyrHeb-16 no exemplo anterior, cores claras não estão mais disponíveis no terminal Linux a menos que seja usado framebuffer.. Se alguém quiser cores claras sem framebuffer e puder viver sem caracteres que não pertencem a seu idioma, ainda é possível usar a fonte 256-glyph, 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 uma autoconversão de keymap de ISO-8859-15 para UTF-8 e habilita deadkeys 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 keymaps tem dead keys (i.e., teclas que não produzem caracteres, mas poem um acento em no caracter produzido pela próxima tecla) ou define uma regra de composição (tal como: “press Ctrl+. A E para ter Æ” no keymap padrão).). Linux-3.13.3 interpreta dead keys e regras de composição no keymap corretamente apenas quando os caracteres fonte a serem compostos não são multibyte. Esta deficiência não afeta keymaps para línguas européias, pois seus acentos são adicionados a caracteres ASCII não acentuados, ou dois caracteres ASCII são compostos em um. Entretanto, em modo UTF-8 isso é um problema, e.g., para a língua grega, onde alguém às vezes precisa colocar um acento na letra “alpha”. A solução pode ser evitar o uso de UTF-8, ou instalar um sistema de janelas X que não tem essa limitação em suas entradas.
- Para chinês, japonês, coreano e algumas outras línguas, o terminal Linux não pode ser configurado para exibir os caracteres necessários. Usuários que precisam de tais idiomas devem instalar um Sistema de Janelas X, fontes que cobrem os caracteres necessários, e o método de entrada adequado (e.g., SCIM, que suporta uma ampla variedade de idiomas).

Nota

O arquivo `/etc/sysconfig/console` apenas contra a localização do terminal de texto do Linux. Ele não tem nada a ver com configurações sobre o layout de teclado adequado e fontes no Sistemas de Janelas X, em sessões ssh ou em um terminal serial. Em tais situações, as limitações mencionadas nos últimos dois itens acima não se aplicam.

7.11. Configurando o script sysklogd

O script `sysklogd` chama o programa **syslogd** com a opção `-m 0`. Esta opção desativa a marcação periódica de tempo do **syslogd** que faz os registros nos arquivos de log a cada 20 minutos por padrão. Se você quiser habilitar este modo, edite o script `/etc/sysconfig/rc.site` e defina a variável SYSKLOGD_PARMS com o valor desejado. Por exemplo, para remover todos os parâmetros, configure a variável para um valor null:

```
SYSKLOGD_PARMS=
```

Veja **`man syslogd`** para mais opções.

7.12. O Arquivo rc.site

O arquivo opcional `/etc/sysconfig/rc.site` contém configurações que são automaticamente configuradas para cada script de boot. Ele pode alternativamente configurar os valores especificados nos arquivos `hostname`, `console`, e `clock` no diretório `/etc/sysconfig/`. Se as variáveis associadas estão presentes em ambos os arquivos e em `rc.site`, s valores no arquivos específicos do script têm precedência.

`rc.site` também contém parâmetros que podem customizar outros aspectos do processo de inicialização. Configurando a variável IPROMPT irá habilitar execução seletiva de bootscripts.. Outras opções são descritas nos comentários nos arquivos. 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}"

# 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
#LOGLEVEL=5

# 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=
```

7.12.1. Customizando os scripts de Inicialização e Desligamento

Os boot scripts do LFS inicializam e desligam o sistema de uma maneira eficiente, mas há alguns ajustes que você pode fazer no arquivo rc.site para melhorar a velocidade e ajustar as mensagens de acordo com suas preferências. Para fazer isso, ajuste as configurações no arquivo `/etc/sysconfig/rc.site` acima.

- Durante o script de boot `udev`, há uma chamada para **udev settle** que requer algum tempo para ser completada. Este tempo pode ou não ser necessário dependendo dos dispositivos presentes no sistema. Se você tem apenas partições simples e uma única placa ethernet, o processo de inicialização provavelmente não precisará esperar por este comando. Para pular isso, ajuste a variável OMIT_UDEV_SETTLE=y.
- O script de boot `udev_retry` também executa **udev settle** por padrão. Este comando é apenas necessário por padrão se o diretório `/var` for montado separadamente. Isso acontece porque o relógio precisa do arquivo `/var/lib/hwclock/adjtime`. Outras customizações também podem precisar esperar que o udev complete, mas em muitas instalações isso não é necessário. Pule o comando ajustando a variável OMIT_UDEV_RETRY_SETTLE=y.
- Por padrão, as checagens do sistema de arquivos são silenciosas. Isso pode parecer um atraso durante o processo de inicialização. Para ativar a saída do **fsck**, ajuste a variável VERBOSE_FSCK=y.
- Quando reinicializando, você pode querer pular a checagem de sistema de arquivos, **fsck**, completamente. Para fazer isso, você pode criar o arquivo `/fastboot` ou reiniciar o sistema com o comando **/sbin/shutdown -f -r now**. Em contra partida, você pode forçar a checagem do sistema de arquivos criando `/forcefsck` ou rodando **shutdown** com o parâmetro `-F` em vez de `-f`.

  Ajustar a variável FASTBOOT=y irá desabilitar **fsck** durante o processo de inicialização até que ele seja removido. Isso não é recomendado como uma abordagem permanente.
- Normalmente, todos os arquivos no diretório `/tmp` são deletados durante o boot. Dependendo do número de arquivos ou diretórios presentes, isso pode causar um atraso notável no processo de boot. Para pular a remoção desses arquivos ajuste a variável SKIPTMPCLEAN=y.
- Durante o desligamento, o programa **init** envia um sinal TERM para cada programa que ele inicializou (e.g. agetty), espera um certo tempo (padrão 3 segundos), e envia a cada processo um sinal KILL e aguarda novamente. Este processo é repetido no script **sendsignals** para quaisquer processos que não sejam desligados por seus próprios scripts. O atraso para **init** pode ser ajustado passando um parâmetro. Por exemplo para remover o atraso em **init**, passe o parâmetro -t0 quando desligando ou reinicializando (e.g. **/sbin/shutdown -t0 -r now**). O atraso para o script **sendsignals** pode ser pulado ajustando o parâmetro KILLDELAY=0.

7.13. Os Arquivos de inicialização do Bash Shell

O programa de shell **/bin/bash** (daqui por diante chamado de “o shell”) utiliza um conjunto de arquivos de inicialização que auxilia na criação do ambiente interativo. Cada arquivo tem um uso específico e pode afetar o login e o ambiente de trabalho de formas diferentes. Os arquivos no diretório `/etc` fornecem ajustes globais. Se um arquivo equivalente existir no diretório home do usuário, suas definições podem se sobrepor aos ajustes globais.

Um ambiente interativo login-shell é iniciado após um login bem sucedido, usando o **/bin/login**, lendo o arquivo `/etc/passwd`. Um ambiente non-login-shell é iniciado na linha de comando (por exemplo, `[prompt]$`**/bin/bash**). Um shell não-interativo está geralmente presente quando um shell script está sendo executado Ele é não-interativo porque está processando um script e não esperando pela entrada de um comando do usuário.

Para mais informações, veja **info bash** na seção *Arquivos de Inicialização do Shell 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` bbásico configurado abaixo, define algumas variáveis de ambiente necessárias para o suporte à língua nativa. Ajustá-lo corretamente resulta em:

- A saída dos programas traduzidos para a língua nativa
- Classificação correta dos caracteres em letras, em dígitos e em outras classes. Isto é necessário para o **bash** para aceitar corretamente caracteres de não-ASCII em linhas de comando em locales não-Ingleses
- A ordem de classificação alfabética correta para o país
- Tamanho do papel padrão apropriado
- Formato correta monetário, de tempo, e de datas

Substitua o *<ll>* nos comandos abaixo pelo código de duas letras correspondente à língua desejada (por exemplo, “en”) e *<CC>* com o código de duas letras correspondente ao seu país (por exemplo, “GB”). *<charmap>* deve ser substituído pelo charmap canônico do locale escolhido. Modificadores opcionais tais como “@euro” podem também estar presente.

A lista de todos os locales suportados pela Glibc pode ser obtida executando o seguinte comando:

```
locale -a
```

Charmaps podem ser um número ou apelido, e.g., “ISO-8859-1” também é referenciado como “iso8859-1” e “iso88591”. Algumas aplicações não podem manipular os vários sinônimos corretamente (e.g., requer que “UTF-8” seja escrito como “UTF-8”, não “utf8”), assim é o mais seguro escolher o nome canônico para um locale particular. Para determinar o nome canônico, execute o seguinte comando, onde *<locale name>* é a saída dada pelo **locale -a**para seu locale preferido (“en_GB.iso88591” ino nosso exemplo).

```
LC_ALL=<locale name> locale charmap
```

Para o locale “en_GB.iso88591”, o comando acima irá imprimir:

```
ISO-8859-1
```

Isto finalmente resulta em um locale do tipo “en_GB.ISO-8859-1”. É importante que o locale encontrado usando a heurística acima seja testado antes que 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 devem imprimir o nome da língua, a codificação de caracteres usada pelo locale, a moeda local, e o prefixo para discar antes de um número de telefone para ligar para o país. Se quaisquer dos comandos acima falhar com uma mensagem similar àquela mostrada a baixo, isso significa que seu locale ou não foi instalada no Capítulo 6 ou não é suportada pela instalação padrão do Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Se isso acontecer, você deve instalar a locale desejada com o comando **localedef**, ou considere escolher um locale diferente. Instruções posteriores assumem que não existam tais mensagens de erro do Glibc.

Alguns pacotes fora do LFS podem não ter suporte ao seu locale escolhido. Um exemplo é a biblioteca X (parte do X Window System), que exibe a seguinte mensagem de erro se o locale não corresponde exatamente a um dos nomes de mapas de caracteres em seus arquivos internos:

```
Warning: locale not supported by Xlib, locale set to C
```

Em vários casos Xlib espera que o mapa de caracteres seja listado em notação de caixa alta com traços canônicos. Por exemplo, "ISO-8859-1" em vez de "iso88591". É também possível encontrar uma especificação apropriada removendo a parte do mapa de caracteres da especificação da locale. Isso pode ser verificado rodando o comando **locale charmap** em ambos locales. por exemplo, alguém poderia ter que mudar "de_DE.ISO-8859-15@euro" para "de_DE@euro" para ter esse locale reconhecido pelo Xlib.

Outros pacotes também podem funcionar incorretamente (mas podem não necessariamente exibir quaisquer mensagens de erro) se o nome do locale não corresponder às expectativas. Nesses casos, investigar como outras distribuições Linux suportam seu locale pode dispor algumas informações úteis.

Uma vez que a configuração de locale adequada tenha sido determinada, 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” (o recomendado para usuários de United States English) são diferentes. “C” usa o conjunto de caracteres US-ASCII 7-bit, e trata bytes com o bit mais alto presente como caracteres inválidos. Esse é o porquê, e.g., do comando **ls** substituí-los com ponto de interrogação neste locale. Também, uma tentativa de enviar emails com tais caracteres com o Mutt ou Pine resulta em mensagens de non-RFC-conforming sendo enviadas (o conjunto de caracteres no email enviado é indicado como “unknown 8-bit”). Então você pode usar o locale “C” apenas se você tiver certeza de que nunca precisará de caracteres 8-bit.

Locales baseados em UTF-8 não são bem suportados por muitos programas. Trabalho está em progresso para documentar e, se possível, consertar tais problemas, veja *http://www.linuxfromscratch.org/blfs/view/svn/introduction/locale-issues.html*.

7.14. Criando o Arquivo /etc/inputrc

O arquivo `inputrc` manipula mapas de teclados para situações específicas. Ele é o arquivo de inicialização usado pelo Readline — biblioteca relacionada ao input — utilizado pelo Bash e pela maioria dos pacotes shell.

A maioria das pessoas não necessita de um mapeamento de teclado específico para o usuário, assim, o comando abaixo cria um `/etc/inputrc` global utilizado por todos que façam logon no sistema. Se você mais tarde precisar substituir o padrão por uma base por usuário, você pode criar um arquivo `.inputrc` no diretório home do usuário com o mapeamento modificado.

Para mais informação sobre como editar o arquivo `inputrc`, veja **info bash** sob a seção *Readline Init File*. **info readline** é também uma boa fonte de info.

Segue abaixo um `inputrc` global e genérico junto com os comentários explicativos das várias opções. Note que os comentários não podem estar na mesma linha que os comandos. Crie o arquivo a 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
```

Capítulo 8. Tornando o Sistema LFS inicializável

8.1. Introdução

É hora de fazer o sistema LFS inicializável. Esse capítulo discute a criação de um arquivo `fstab`, compilação de um kernel para o novo sistema LFS, e instalação do gerenciador de inicialização GRUB de modo que o sistema LFS possa ser selecionado para boot durante a inicialização.

8.2. Criando o arquivo /etc/fstab

O arquivo `/etc/fstab` é usado por alguns programas para determinar onde arquivos de sistema devem ser montados por padrão, em que ordem, e quais devem ser checados (por erros de integridade) andes de serem montados. Crie uma nova tabela de sistemas de arquivos dessa forma:

```
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>* om os valores apropriados para o sistema, por exemplo, `sda2`, `sda5`, e `ext4`. Para detalhes sobre os seis campos nesse arquivo, veja **man 5 fstab**.

Sistema de arquivo com origem no MS-DOS ou Windows (i.e.: vfat, ntfs, smbfs, cifs, iso9660, udf) precisam da opção de montagem “iocharset” para que os caracteres não-ASCII nos nomes dos arquivos sejam interpretados corretamente. O valor dessa opção deve ser o mesmo que o conjunto de caracteres de seu locale, ajustado de tal forma que o kernel o entenda. Isso funciona de a definição de charset relevante (encontrado em File systems -> Native Language Support) foi compilada no kernel ou construída como um módulo. A opção “codepage” também é necessária para sistemas de arquivos vfat e smbfs. Deve ser configurado para o número codepage usado no MS-DOS em seu país. E.g., para montar flash drives USB, um usuário ru_RU.KOI8-R precisaria do seguinte na parte de opções do seu mount em `/etc/fstab`:

```
noauto,user,quiet,showexec,iocharset=koi8r,codepage=866
```

O fragmentos de opções correspondente para o usuário ru_RU.UTF-8 é:

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

Nota

Neste último caso, o kernel emit a seguinte mensagem:

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
    filesystem will be case sensitive!
```

Essa recomendação negative deve ser ignorada, uma vez que todas as outras opções de “iocharset” resultam em exibição errada dos nomes de arquivos em locales UTF-8.

É também possível especificar codepage padrão e valores iocharset para alguns sistemas de arquivos durante a configuração do kernel. Os parâmetros relevantes são nomeados “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 fazer o sistema de arquivos et3 confiável em casos de falha de energia 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 drive de disco suporta essa opção, execute *hdparm* no drive de disco aplicável. Por exemplo, se:

```
hdparm -I /dev/sda | grep NCQ
```

retorna uma saída não-vazia, a opção é suportada.

Nota: partições baseadas em Logical Volume Management (LVM) não podem usar a opção `barrier`.

8.3. Linux-3.13.3

O pacote Linux contém o Kernel Linux

Tempo de Construção: 3.0 - 49.0 SBU (tipicamente cerca de 6 SBU)
Espaço de disco: 700 - 6800 MB (tipicamente entre 800-900 MB)

8.3.1. Instalação do Kernel

Construir o kernel envolve alguns passos—configuração, compilação, e instalação. Leia o arquivo README no código fonte do kernel para métodos alternativos às maneiras que este livro configura o kernel.

Prepare para compilação corando o seguinte comando:

```
make mrproper
```

Isso garante que a árvore do kernel esteja absolutamente limpa. O time do kernel recomenda que este comando seja executado antes de cada compilação do kernel. Não confie que o código fonte esteja limpo após descompatar.

Configure o kernel através de uma interface baseada em menus. Para informações gerais sobre configuração do kernel veja *http://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt*. BLFS tem algumas informações relativas a configurações particulares do kernel necessárias para pacotes que estão fora do LFS em *http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index*. Informação adicional sobre configurar e cosntruir o kernel pode ser encontrada em *http://www.kroah.com/lkn/*

Nota

Uma boa maneira de começar a definir a configuração do kernel é executando **make defconfig**. Isso irá definir a configuração base para um bom estado que se adeque a sua máquina.

Devido a mudanças recentes no udev, certifique-se de selecionar:

```
Device Drivers  --->
  Generic Driver Options  --->
     Maintain a devtmpfs filesystem to mount at /dev
```

```
make LANG=<host_LANG_value> LC_ALL= menuconfig
```

O significado dos parâmetros do make:

LANG=<host_LANG_value> LC_ALL=

Isso estabelece a cofniguração do locale para aquela usada no sistema anfitrião. Isso é necessário para a renderização adequada de linhas ncurse do menuconfig em consoles de texto UTF-8.

Certifique-se de substituir *<host_LANG_value>* pelo valor da variável $LANG do seu sistema anfitrião. Se não configurada, você poderia usar em vez disso o valor da variável $LC_ALL ou $LC_CTYPE.

Alternativamente, **make oldconfig** pode ser mais apropriado em algumas situações. Veja o arquivo README para mais informações.

Se desejar, pule a configuração do kernel copiando o arquivo, `.config` do sistema anfitrião (assumindo que ele esteja disponível) para o diretório `linux-3.13.3`. Entertanto, nós não recomendamos essa opção. Geralmente 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 estiver usando módulos do kernel, configuração do módulo em `/etc/modprobe.d` pode ser necessária. Informações pertinentes à configuração dos módulos e do kernel estão localizadas em Seção 7.4, “Manipulando dispositivos e módulos em um Sistema LFS” e na documentação do kernel no diretório `linux-3.13.3/Documentation`. Além disso, `modprobe.conf(5)` pode ser de interesse.

Instale os módulos, se a configuração do kernel usa os mesmos:

```
make modules_install
```

Após a compilação do kernel completa, passos adicionais são necessários para completar a instalação. Alguns arquivos precisam ser copiados para o diretório `/boot`.

O caminho para a imagem do kernel pode varir dependendo da plataforma usada. O nome de arquivo abaixo pode ser alterado para se adequar a seu gosto, mas a base deve ser *vmlinuz* para ser compatível com a configuração automática do processo de boot descrito na próxima seção. O seguinte comando assume que uma arquitetura x86 está em uso:

```
cp -v arch/x86/boot/bzImage /boot/vmlinuz-3.13.3-lfs-7.5
```

`System.map` é um arquivo simbolo para o kernel. Ele mapeia os pontos de entrada de função de cada função da API do Kernel, assim como os endereços das estruturas de dados do kernel para o kernel em execução. É usado como um recurso quando investigando problemas no kernel. Execute o seguinte comando para instalar o arquivo map:

```
cp -v System.map /boot/System.map-3.13.3
```

O arquivo de configuração do kernel `.config` produzido pelo passo **make menuconfig** acima contém todas as seleções de configurações para o kernel que acabou de ser compilado. É uma boa idéia manter esse arquivo para referências futuras:

```
cp -v .config /boot/config-3.13.3
```

Instale a documentação para o kernel Linux:

```
install -d /usr/share/doc/linux-3.13.3
cp -r Documentation/* /usr/share/doc/linux-3.13.3
```

É importante notar que os arquivos no diretório fonte do kernel não pertencem ao *root*. Sempre que um pacote é desempacotado como um usuário *root* (como nós fizemos dentro do chroot), os arquivos tem os IDs de grupo e usuário que eles tinham no computador de quem empacotou. Isso geralmente não é um problema para qualquer outro pacote a ser instalado porque a árvore do código fonte é removida depois da instalação. Entretando, o código fonte do Linux é geralmente mantido por um longo tempo. Devido a isso, há uma chance de qualquer usuário do ID de usuário do empacotador ser usado por alguém na máquina. Essa pessoa teria então acesso de escrita ao código fonte do kernel.

Se o código fonte do kernel for mantido, rode **chown -R 0:0** no diretório `linux-3.13.3` para assegurar que todos os arquivos pertençam ao usuário *root*.

Atenção

Algumas documentações do kernel recomendam criar um symlink `/usr/src/linux` apontando para o diretório do código fonte do kernel. Isso é específico para kernel anterior a série 2.6 e *must not* ser criado em um sistema LFS uma vez que pode causar problemas para pacotes que você pode querer construir uma vez que seu sistema LFS base esteja completo.

Atenção

Os cabeçalhos no diretório `include` do sistema (`/usr/include`) devem *always* ser aqueles com os quais Glibc foi compilada, isto é, os headers limpos instalados no Seção 6.7, “Linux-3.13.3 API Headers”. Portanto, eles *nunca* devem ser substituídos tanto por cabeçalhos crus do kernel ou quaisquer outros cabeçalhos limpos de kernel.

8.3.2. Configurando Ordem de Carregamento de Módulos do Linux

Na maioria das vezes módulos Linux são carregados automaticamente, mas algumas vezes precisa-se de alguma instrução específica. O programa que carrega módulos, **modprobe** ou **insmod**, usa `/etc/modprobe.d/usb.conf` para esse propósito. Esse arquivo precisa ser criado para o caso de drivers USB (ehci_hcd, ohci_hcd and uhci_hcd) terem sido construídos como módulos, eles serão carregados na ordem correta; ehci_hcd precisa ser carregado antes de ohci_hcd e uhci_hcd para evitar que um alerta(warning) seja exibido em tempo de boot.

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
```

8.3.3. Conteúdo do Linux

Arquivos instalados: config-3.13.3, vmlinuz-3.13.3-lfs-7.5, and System.map-3.13.3
Diretórios instalado: /lib/modules, /usr/share/doc/linux-3.13.3

Breve descrição

`config-3.13.3` Contém todas as seleções de configurações para o kernel

`vmlinuz-3.13.3-lfs-7.5` O núcleo do sistema Linux. Quando ligando o computador, o kernel é a primeira parte do sistema operacional a ser carregada. Ele detecta e inicializa todos os componentes de hardware do computador, então torna esses componentes disponívies como uma árvore de arquivos para o software e torna uma única CPU em uma máquina multitarefa capaz de rodar uma multidão de programas aparentemente ao mesmo tempo

`System.map-3.13.3` Uma lista de endereços e símbolos; ele mapeia os pontos de entrada e endereços de todas as funções e estruturas de dados no kernel

8.4. Usando o GRUB para Configurar o Processo de Boot

8.4.1. Introdução

Atenção

Configurar o GRUB incorretamente pode tornar seu sistema inoperável sem um sistema de boot alternativo como um CD-ROM. Esta seção não é necessária para inicializar seu sistema LFS. Você pode apenas querer modificar seu gerenciador de boot atual, e.g. Grub-Legacy, GRUB2, ou LILO.

Certifique-se que um disco de boot de emergência esteja pronto para “rescue” to computador se o computador se tornar inútil (un-bootable). Se você ainda não tem um dispositivo de boot, você pode criar um. Para que o procedimento abaixo funcione, você precisa pular para o BLFS e instalar **`xorriso`**do pacote *libisoburn*.

```
cd /tmp &&
grub-mkrescue --output=grub-img.iso &&
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

8.4.2. Convenções para Nomenclatura do GRUB

GRUB usa sua própria estrutura de nomes para dispositivos e partições na forma de *(hdn,m)*, onde *n* é o número do disco rígido e *m* é o número da partição. O número do disco rígindo começa do zero, mas o número da partição inicia do um para partições normais e cinco para partições extendidadas. Note que isso é diferente de versões anteriores onde ambos números começavam do zero. Por exemplo, partição `sda1` é *(hd0,1)* para o GRUB e `sdb3` é *(hd1,3)*. Em contraste com o Linux, GRUB não considera drives de CD-ROM como discos rígidos. Por exemplo, se usar um CD em `hdb` e um segundo disco rígido em `hdc`, aquele segundo disco rígido ainda seria *(hd1)*.

8.4.3. Definindo a Configuração

GRUB funciona escrevendo dados na primeira faixa física do disco rígido. Essa área não é parte de nenhum sistema de arquivos. Os programas lá acessam módulos do GRUB na partição de boot. A localização padrão é /boot/grub/.

A localização da partição de boot é uma escolha do usuário que ageta a configuração. Uma recomendação é ter uma partição extra pequena (tamanho sugerido é 100 MB) apenas para informação de boot. Dessa forma cada construção, seja LFS ou aguma distro comercial, pode acessar os mesmos arquivos de boot e o acesso pode ser feito de qualquer sistema inicializado. Se você escolher fazer isso, você precisará montar a partição separada, mover todos os arquivos no diretório `/boot` atual (e.g. o kernel linux que você acabou de construir na seção anterior) para a nova partição. Você precisará então desmontar a partição e remontar como `/boot`. Esses nomes serão necessários posteriormente para o arquivo `/etc/fstab`.

Usar a partição lfs atual também funcionará, mas configuração para múltiplos sistemas é mais difícil.

Usando a informação acima, determina o designador apropriado para a partição raiz (ou partição de boot, se uma partição separada foi usada). Para o exemplo seguinte, assume-se que a partição raiz (ou boot separado) é `sda2`.

Instale arquivos do GRUB em `/boot/grub` e configure a trilha de inicialização:

Atenção

O seguinte comando irá sobrescrever o gerenciador de boot atual. Não execute o comando de isso não é desejado, por exemplo, se estiver usando um gerenciador de boot de terceiro para gerir o Master Boot Record (MBR).

```
grub-install /dev/sda
```

8.4.4. Criando o Arquivo de Configuração

Gere `/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 3.13.3-lfs-7.5" {
        linux   /boot/vmlinuz-3.13.3-lfs-7.5 root=/dev/sda2 ro
}
EOF
```

Nota

Da perspectiva do GRUB, os arquivos do kernel são relativos à partição usada. Se você usou uma partição /boot separada, remova /boot da linha acima *linux*. Você também precisará mudar a linha *set root* para apontar para a partição de boot.

GRUB é um programa extremamente poderoso e dispõe um tremendo número de opções para inicializar uma variedade de dispositivos, sistemas operacionais e tipos de partição. Há também muitas opções para customização tais como splash screens gráficas, execução de sons, entrada do mouse, etc. Os detalhes dessas opções estão além do escopo dessas instruções.

Cuidado

Há um comando, grub-mkconfig, que pode escrever um arquivo de configuração automaticamente. Ele usa um conjunto de scripts em /etc/grub.d/ e irá destruir quaisquer customizações que você fizer. Esses scripts são desenvolvidos principalmente para distribuições não baseadas em código fonte e não são recomendados para o LFS. Se você instalar uma distribuição Linux comercial, há uma boa chance de esse programa ser executado. Certifique-se de fazer backup de ser arquivo grub.cfg.

Capítulo 9. O final

9.1. O final

Muito bem! O novo sistema LFS está instalado! Nós desejamos a você muito sucesso com seu novo sistema Linux customizado.

Pode ser uma boa idéia criar um arquivo `/etc/lfs-release`. Tento esse arquivo, fica muito fácil pra você (e para nós se você precisar pedir por ajuda em algum ponto) descobrir qual versão LFS está instalada em seu sistema. Crie esse arquivo rodando:

```
echo 7.5 > /etc/lfs-release
```

É também uma boa idéia criar um arquivo para mostrar o status de seu novo sistema no que diz respeito ao Linux Standards Base (LSB). Para criar esse arquivo, rode:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="7.5"
DISTRIB_CODENAME="<your name here>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

Certifique-se de colocar alguma forma de customização para o campo 'DISTRIB_CODENAME' fazer do sistema unicamente seu.

9.2. Seja Contado

Agora que você terminou o livro, você quer ser contato como um usuário LFS? Siga para *http://www.linuxfromscratch.org/cgi-bin/lfscounter.php* e registre-se como um usuário LFS fornecendo seu nome e a primeira versão do LFS que você usou.

Vamos reiniciar no LFS agora.

9.3. Reiniciando o Sistema

Agora que todo o software foi instalado, é hora de reiniciar seu computador. Entretanto, você deve estar ciente de algumas coisas. O sistema que você criou neste livro é mínimo, e provavelmente não tem a funcionalidade que você precisaria para ser capaz de seguir em frente. Instalando alguns pacotes extra do livro BLFS enquanto ainda em seu ambiente chrrot atual, você pode deixar-se em uma posição muito melhor para continuar uma vez que você reinicie em sua nova instalação LFS. Aqui estão algumas sugestões:

- Um browser de texto como o *Lynx* permitirá que você visualize facilmente o livro BLFS em um terminal virtual, enquanto construindo pacotes em outro.
- O pacote *GPM* permitirá que você execute ações de copiar/colar em seus terminais virtuais.
- Se você está em uma situação onde configuração de IP estático não é o caso de suas configurações de rede, instale pacotes tais como *dhcpcd* ou a porção cliente de *dhcp* pode ser útil.
- Instalar *sudo* pode ser útil para construir pacotes como um usuário não-root e facilmente instalar os pacotes resultantes em seu sistema.

- Se você quer acessar seu novo sistema a partir de um sistema remoto dentro de uma GUI confortável, instale *openssh* e seu pré-requisito, *openssl*.
- Para baixar arquivos da internet com mais facilidade, instale *wget*.
- Se um ou mais de seus drives de disco tem uma tabela de partição GUID (GPT), tanto *gptfdisk* ou *parted* serão úteis.
- Finalmente, uma revisão dos arquivos de configuração seguintes é também 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/network
  - /etc/sysconfig/ifconfig.eth0

Agora que nós dissemos isso, vamos seguir para inicialização da nossa novíssima instalação 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
```

Desmonte o sistema LFS propriamente dito:

```
umount -v $LFS
```

Se múltiplas partições foram criadas, desmonte as outras partições antes de desmontar a principal, dessa forma:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Agora, reinicie o sistema com:

```
shutdown -r now
```

Assumindo que o gerenciador de boot GRUB foi configurado como destacado anteriormente, o menu está configurado para iniciar *LFS 7.5* automaticamente.

Quando a reinicialização estiver completa, o sistema LFS estará pronto para uso e mais softwares podem ser adicionados para suprir suas necessidades.

9.4. O que fazer 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 deve estar se perguntando “O que eu faço agora?” Para responder essa questão, nós compilamos uma lista de recursos para você.

- Manutenção

  Bugs e notificações de segurança são relatadas regularmente para todos os softwares. Uma vez que o sistema LFS é compilado do fonte, é responsabilidade sua se manter bem informado sobre esses boletins. Há várias fontes online de recursos que acompanham esses boletins, algumas das uais são mostradas abaixo.

  - Freecode (*http://freecode.com/*)

    Freecode pode notificar você (via email) sobre novas versões dos pacotes instalados no seu sistema.

  - *CERT* (Computer Emergency Response Team)

    CERT tem uma lista de email que publica alertas de segurança sobre vários sistemas operacionais e aplicações. Informação para se subescrever está disponível em *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq é uma lista de email sobre segurança de computadores completa. Ela publica problemas se segurança descobertos recentemente, e ocasionalmente consertos potenciais para eles. Informação para se subescrever 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 softwares além do escopo do livro LFS. O projeto BLFS está localizado em *http://www.linuxfromscratch.org/blfs/*.

- LFS Hints

  Os LFS Hinsts são uma coleção de documentos educacionais submetidos por voluntários na comunidade LFS. As dicas (The hints) estão disponíveis em *http://www.linuxfromscratch.org/hints/list.html*.

- Lista de Email

  Há várias listas de email do LFS nas quais você pode se inscrever se você precisar de ajuda, quiser se manter atualizado com os últimos desenvolvimentos, quiser constribuir com o projeto, e mais. Veja Chapter 1 - Mailing Listspara mais informação.

- The Linux Documentation Project

  O Objetivo do The Linux Documentation Project (TLDP) é colaborar em todas as questões de documentação Linux. O TLDP caracteriza-se por um grande coleção de HOWTOs, guias, e páginas de manuais. It is located at *http://www.tldp.org/*.

Parte IV. Apêndices

Apêndice A. Acrônimos e Abreviaturas

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. Agradecimentos

Nós gostaríamos de agradecer as seguintes pessoas e organizações por suas contribuições ao Projeto Linux From Scratch.

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Criador do LFS, Líder do Projeto LFS
- *Matthew Burgess* <matthew@linuxfromscratch.org> – LFS Project Leader, LFS Technical Writer/Editor
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Gestor de Releases do LFS, Escritor/Editor Técnico do LFS
- *Jim Gifford* <jim@linuxfromscratch.org> – Co-Líder do Projeto CLFS
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – Escritor Técnico do LFS
- *Randy McMurchy* <randy@linuxfromscratch.org> – Líder do Projeto BLFS, Editor do LFS
- *DJ Lucas* <dj@linuxfromscratch.org> – Editor do LFS e BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Editor do LFS e CLFS
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Co-Líder do Projeto CLFS
- Um número incontável de pessoas nas listas de discussão LFS e BLFS que ajudaram a tornar este livro possível dando sugestões, testando o livro, e submetendo relatórios de falhas, instruções e suas experiências com a instalação dos vários pacotes.

Tradutores

- *Manuel Canales Esparcia* <macana@macana-es.com> – Projeto de tradução para espanhol
- *Johan Lenglet* <johan@linuxfromscratch.org> – Projeto de Tradução para francês
- *Alberto Senna Dias Neto* <albertosdneto@gmail.com> – Projeto de tradução para português
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Projeto de tradução para português
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Projeto de tradução para alemão

Mantenedores de espelhos

Espelhos na 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 na 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 Baqué* <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

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 Australianos

- *Jason Andrade* <jason@dstc.edu.au> – espelho au.linuxfromscratch.org

Antigos membros do Projeto

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – Editor do LFS
- Archaic <archaic@linuxfromscratch.org> – Escritor/Editor Técnico do LFS, Líder do Projeto HLFS, Editor do BLFS, Mantenedor do Projeto Dicas e Patches (Hints and Patches)
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Mantenedor do LFS-Bootscripts
- Timothy Bauscher
- Robert Briggs
- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Desenvolvedor do Website, Mantenedor do FAQ
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Mantenedor do 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 do LFS LiveCD
- Mark Hymers
- Seth W. Klein – Mantenedor do FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Mantenedor do Wiki
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Mantenedor dos Backend-Scripts do Website

- *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 do LFS Live CD
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Mantenedor do NNTP Gateway LFS
- *Greg Schafer* <gschafer@zip.com.au> – Escritor Técnico do LFS e Arquiteto do método de construção para habilitação de 64-bit.
- Jesse Tie-Ten-Quee – Escritor Técnico do LFS
- *James Robertson* <jwrober@linuxfromscratch.org> – Mantenedor do Bugzilla
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Editor do BLFS, Líder do Projeto Dicas e Patches (Hints and Patches)
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Escritor Técnico do LFS, Mantenedor do Bugzilla, Mantenedor dos LFS-Bootscripts
- *Zack Winkles* <zwinkles@gmail.com> – Escritor Técnico do LFS

Apêndice C. Dependências

Todo pacote construído no LFS depende de um ou mais pacotes para compilar e instalar propriamente. Alguns pacotes até fazem parte de dependências circulares, ou seja, o primeiro pacote depende do segundo que em retorno depende do primeiro. Por causa dessas dependências, a ordem na qual pacotes sõ construídos no LFS é importante. O propósito desta página é documentar a dependência de cada pacote construído no LFS.

Para cada pacote que nós compilamos, nós listamos três, e às vezes quatro, tipos de dependências. A primeira lista que outros pacotes precisam estar disponíveis para compilar e instalar o pacote em questão. A segunda lista que pacotes, em adição àqueles na primeira lista, precisam estar disponíveis para rodar a suite de testes. A terceira lista de dependências são pacotes que precisam que este pacote esteja compilado e instalado em sua localização final antes que eles sejam compilados e instalados. Em muitos casos, isso acontece porque esses pacotes irão gravar caminhos para binários dentro de seus scripts. Se não for construído em determinada ordem, isso poderia resultar em caminhos como /tool/bin/[binario] sendo colocados dentro de scripts para a instalação final. Isso obviamente não é desejável.

A última lista de dependências são pacotes opcionais que não são discutidos no LFS, mas poderiam ser úteis ao usuário. Esses pacotes podem ter dependências adicionais obrigatórias ou opcionais. Para essas dependências, a prática recomendada é instalá-las depois de completar o livro LFS e então voltar e reconstruir o pacote LFS. Em muitos casos a reinstalação é discutida no BLFS.

Autoconf

A instalação depende de: Bash, Coreutils, Grep, M4, Make, Perl, Sed, e Texinfo

Suite de testes depende de: Automake, Diffutils, Findutils, GCC, e Libtool

Tem que ser instalado antes de : Automake

Dependências opcionais: Emacs

Automake

A instalação depende de: Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed, e Texinfo

Suite de testes depende de: Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool, e Tar.

Tem que ser instalado antes de : None

Dependências opcionais: None

Bash

A instalação depende de: Bash, Binutils, Bison, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed, e Texinfo

Suite de testes depende de: Shadow

Tem que ser instalado antes de : None

Dependências opcionais: Xorg

Bc

A instalação depende de: Bash, Binutils, Bison, Coreutils, GCC, Glibc, Grep, Make, e Readline

Suite de testes depende de: Gawk

Tem que ser instalado antes de : Linux Kernel

Dependências opcionais: None

Binutils

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, File, Gawk, GCC, Glibc, Grep, Make, Perl, Sed, Texinfo e Zlib

Suite de testes depende de: DejaGNU e Expect

Tem que ser instalado antes de : None

Dependências opcionais: None

Bison

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, e Sed

Suite de testes depende de: Diffutils, Findutils, e Flex

Tem que ser instalado antes de : Kbd e Tar

Dependências opcionais: Doxygen (suite de testes)

Bzip2

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make, e Patch

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Check

A instalação depende de: GCC, Grep, Make, Sed, e Texinfo

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Coreutils

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, GMP, Grep, Make, Patch, Perl, Sed, e Texinfo

Suite de testes depende de: Diffutils, E2fsprogs, Findutils, Shadow, e Util-linux

Tem que ser instalado antes de : Bash, Diffutils, Findutils, Man-DB, e Udev

Dependências opcionais: Perl Expect and IO:Tty modules (para suite de testes)

DejaGNU

A instalação depende de: Bash, Coreutils, Diffutils, GCC, Grep, Make, e Sed

Suite de testes depende de:

Tem que ser instalado antes de : None

Dependências opcionais: None

Diffutils

A instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Diffutils, Perl

Tem que ser instalado antes de : None

Dependências opcionais: None

Expect

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed, e Tcl

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

E2fsprogs

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Gzip, Make, Sed, Texinfo, e Util-linux

Suite de testes depende de: Procps-ng,Psmisc

Tem que ser instalado antes de : None

Dependências opcionais: None

File

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Zlib

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Findutils

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: DejaGNU, Diffutils, e Expect

Tem que ser instalado antes de : None

Dependências opcionais: None

Flex

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed, e Texinfo

Suite de testes depende de: Bison (suppressed) e Gawk

Tem que ser instalado antes de : IPRoute2, Kbd, e Man-DB

Dependências opcionais: None

Gawk

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed e, Texinfo

Suite de testes depende de: Diffutils

Tem que ser instalado antes de : None

Dependências opcionais: None

Gcc

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, GMP, Grep, M4, Make, MPC, MPFR, Patch, Perl, Sed, Tar, e Texinfo

Suite de testes depende de: DejaGNU e Expect

Tem que ser instalado antes de : None

Dependências opcionais: *CLooG-PPL*, *GNAT* and *PPL*

GDBM

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Grep, Make, e Sed

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Gettext

A instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Diffutils, Perl, e Tcl

Tem que ser instalado antes de : Automake

Dependências opcionais: None

Glibc

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Linux API Headers, Make, Perl, Sed, e Texinfo

Suite de testes depende de: File

Tem que ser instalado antes de : None

Dependências opcionais: None

GMP

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, M4, Make, Sed e Texinfo

Suite de testes depende de: None

Tem que ser instalado antes de : MPFR, GCC

Dependências opcionais: None

Grep

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed, e Texinfo

Suite de testes depende de: Gawk

Tem que ser instalado antes de : Man-DB

Dependências opcionais: Pcre, Xorg, e CUPS

Groff

A instalação depende de: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed, e Texinfo

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Man-DB e Perl

Dependências opcionais: GPL Ghostscript

GRUB

A instalação depende de: Bash, Binutils, Bison, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Texinfo, e Xz

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Gzip

A instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Diffutils, Less

Tem que ser instalado antes de : Man-DB

Dependências opcionais: None

Iana-Etc

A instalação depende de: Coreutils, Gawk, e Make

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Perl

Dependências opcionais: None

Inetutils

A instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, Texinfo, e Zlib

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Tar

Dependências opcionais: None

IProute2

A instalação depende de: Bash, Bison, Coreutils, Flex, GCC, Glibc, Make, e Linux API Headers

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: None

Kbd

A instalação depende de: Bash, Binutils, Bison, Check, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch, e Sed

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: None

Kmod

A instalação depende de: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Sed, Xz-Utils, Zlib

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Udev

Dependências opcionais: None

Less

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, e Sed

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Gzip

Dependências opcionais: Pcre

Libpipeline

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Check

Tem que ser instalado antes de : Man-DB

Dependências opcionais: None

Libtool

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Findutils

Tem que ser instalado antes de : None

Dependências opcionais: None

Linux Kernel

A instalação depende de: Bash, Bc, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Kmod, Make, Ncurses, Perl, e Sed

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: None

M4

A instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Diffutils

Tem que ser instalado antes de : Autoconf e Bison

Dependências opcionais: libsigsegv

Make

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Perl e Procps-ng

Tem que ser instalado antes de : None

Dependências opcionais: None

Man-DB

A instalação depende de: Bash, Binutils, Bzip2, Coreutils, Flex, GCC, GDBM, Gettext, Glibc, Grep, Groff, Gzip, Less, Libpipeline, Make, Sed, e Xz

Suite de testes depende de: Util-linux

Tem que ser instalado antes de : None

Dependências opcionais: None

Man-Pages

A instalação depende de: Bash, Coreutils, e Make

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: None

MPC

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, MPFR, Sed e Texinfo

Suite de testes depende de: None

Tem que ser instalado antes de : GCC

Dependências opcionais: None

MPFR

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, GMP, Make, Sed e Texinfo

Suite de testes depende de: None

Tem que ser instalado antes de : GCC

Dependências opcionais: None

Ncurses

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch, e Sed

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Bash, GRUB, Inetutils, Less, Procps-ng, Psmisc, Readline, Texinfo, Util-linux, e Vim

Dependências opcionais: None

Patch

A instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, e Sed

Suite de testes depende de: Diffutils

Tem que ser instalado antes de : None

Dependências opcionais: Ed

Perl

A instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, GDBM, Glibc, Grep, Groff, Make, Sed, e Zlib
Suite de testes depende de: Iana-Etc e Procps-ng
Tem que ser instalado antes de : Autoconf
Dependências opcionais: None

Pkg-config

A instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Popt, e Sed
Suite de testes depende de: None
Tem que ser instalado antes de : Kmod
Dependências opcionais: None

Popt

A instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make
Suite de testes depende de: Diffutils e Sed
Tem que ser instalado antes de : Pkg-config
Dependências opcionais: None

Procps-ng

A instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Make, e Ncurses
Suite de testes depende de: DejaGNU
Tem que ser instalado antes de : None
Dependências opcionais: None

Psmisc

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, e Sed
Suite de testes depende de: Nenhuma suite de testes disponível
Tem que ser instalado antes de : None
Dependências opcionais: None

Readline

A instalação depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed, e Texinfo
Suite de testes depende de: Nenhuma suite de testes disponível
Tem que ser instalado antes de : Bash
Dependências opcionais: None

Sed

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed, e Texinfo

Suite de testes depende de: Diffutils e Gawk

Tem que ser instalado antes de : E2fsprogs, File, Libtool, e Shadow

Dependências opcionais: Cracklib

Shadow

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, e Sed

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : Coreutils

Dependências opcionais: Acl, Attr, Cracklib, PAM

Sysklogd

A instalação depende de: Binutils, Coreutils, GCC, Glibc, Make, e Patch

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: None

Sysvinit

A instalação depende de: Binutils, Coreutils, GCC, Glibc, Make, e Sed

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: None

Tar

A instalação depende de: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Sed, e Texinfo

Suite de testes depende de: Autoconf, Diffutils, Findutils, Gawk, e Gzip

Tem que ser instalado antes de : None

Dependências opcionais: None

Tcl

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, e Sed

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Texinfo

A instalação depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, e Sed

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Udev

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Kmod, Make, Sed, e Util-linux

Suite de testes depende de: Nenhuma suite de testes disponível

Tem que ser instalado antes de : None

Dependências opcionais: Glib, Pci-Utils, Python, Systemd, USB-Utils

Util-linux

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Ncurses, Sed, Udev, e Zlib

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: None

Vim

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, e Sed

Suite de testes depende de: None

Tem que ser instalado antes de : None

Dependências opcionais: Xorg, GTK+2, LessTif, Python, Tcl, Ruby, e GPM

Xz

A instalação depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, e Make.

Suite de testes depende de: None

Tem que ser instalado antes de : GRUB, Kmod, Man-DB, Udev

Dependências opcionais: None

Zlib

A instalação depende de: Bash, Binutils, Coreutils, GCC, Glibc, Make, e Sed

Suite de testes depende de: None

Tem que ser instalado antes de : File, Kmod, Perl, e Util-linux

Dependências opcionais: None

Apêndice D. Versão dos scripts de boot e sysconfig-20130821

Os scripts neste apêndice são 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 são listados na ordem em que eles são normalmente chamados.

D.1. /etc/rc.d/init.d/rc

O script `rc` é o primeiro script chamado pelo init e inicia o processo de boot.

```
#!/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 files provided by\n"
   MSG="${MSG}the ${DISTRO_MINI} book, please be so kind to inform us at\n"
   MSG="${MSG}${DISTRO_CONTACT}.\n"
   log_failure_msg "${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."
      continue
   fi
```

```
   if [ ! -x ${i} ]; then
      log_warning_msg "${i} is not executable, skipping."
      continue
   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 /var/run/interactive ] && source /var/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

      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

   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\"" > /var/run/interactive
```

```
else
    rm -f /var/run/interactive 2> /dev/null
fi

# Copy the boot log on initial boot only
if [ "${previous}" == "N" -a  "${runlevel}" != "S" ]; then
   cat /run/var/bootlog >> /var/log/boot.log

   # Mark the end of boot
   echo "--------" >> /var/log/boot.log

   # Remove the temporary file
   rm -f /run/var/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"

## 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"

## 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}"

SUCCESS_SUFFIX="${BRACKET}[${SUCCESS}  OK  ${BRACKET}]${NORMAL}"
FAILURE_SUFFIX="${BRACKET}[${FAILURE} FAIL ${BRACKET}]${NORMAL}"
WARNING_SUFFIX="${BRACKET}[${WARNING} WARN ${BRACKET}]${NORMAL}"

BOOTLOG=/run/var/bootlog
KILLDELAY=3

# Set any user specified environment variables e.g. HEADLESS
[ -r /etc/sysconfig/rc.site ]  && . /etc/sysconfig/rc.site

################################################################################
# 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.
            rm -f "${pidfile}"

            # 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 "/var/run/${progname}.pid" ]; then
                rm -f "/var/run/${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 it's job, and evaluate kills 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 "/var/run/${progname}.pid" ]; then
            pidfile="/var/run/${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 "${base}" -a -e "/var/run/${base}.pid" ]; then
         /bin/echo -e "${WARNING}${1} is not running but" \
            "/var/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_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} | {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 or -{signal} or {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/functions

```
#!/bin/sh
########################################################################
# Begin boot functions
#
# Description : Run Level Control Functions
#
# Authors     : Gerard Beekmans - gerard@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
#
#               This file is only present for backward BLFS compatibility
#
########################################################################

## Environmental setup
# Setup default values for environment
umask 022
export PATH="/bin:/usr/bin:/sbin:/usr/sbin"

# Signal sent to running processes to refresh their configuration
RELOADSIG="HUP"

# Number of seconds between STOPSIG and FALLBACK when stopping processes
KILLDELAY="3"

## 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))

## Provide an echo that supports -e and -n
# If formatting is needed, $ECHO should be used
case "`echo -e -n test`" in
   -[en]*)
      ECHO=/bin/echo
      ;;
   *)
      ECHO=echo
      ;;
esac

## 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

## 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

STRING_LENGTH="0"   # the length of the current message

#*************************************************************************
# Function - boot_mesg()
#
# Purpose:      Sending information from bootup scripts to the console
#
# Inputs:       $1 is the message
#               $2 is the colorcode for the console
#
# Outputs:      Standard Output
#
# Dependencies: - sed for parsing strings.
#          - grep for counting string length.
#
# Todo:
#*************************************************************************
boot_mesg()
{
   local ECHOPARM=""

   while true
   do
```

```
      case "${1}" in
         -n)
            ECHOPARM=" -n "
            shift 1
            ;;
         -*)
            echo "Unknown Option: ${1}"
            return 1
            ;;
         *)
            break
            ;;
      esac
   done

   ## Figure out the length of what is to be printed to be used
   ## for warning messages.
   STRING_LENGTH=$((${#1} + 1))

   # Print the message to the screen
   ${ECHO} ${ECHOPARM} -e "${2}${1}"

   # Log the message
   [ -d /run/var ] || return
   ${ECHO} ${ECHOPARM} -e "${2}${1}" >> /run/var/bootlog
}

boot_mesg_flush()
{
   # Reset STRING_LENGTH for next message
   STRING_LENGTH="0"
}

echo_ok()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${SUCCESS}  OK  ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ OK ]" >> /run/var/bootlog
}

echo_failure()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${FAILURE} FAIL ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
    boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ FAIL]"  >> /run/var/bootlog
}

echo_warning()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} WARN ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
```

```
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "[ WARN ]"  >> /run/var/bootlog
}

echo_skipped()
{
   ${ECHO} -n -e "${CURS_UP}${SET_COL}${BRACKET}[${WARNING} SKIP ${BRACKET}]"
   ${ECHO} -e "${NORMAL}"
   boot_mesg_flush

   [ -d /run/var ] || return
   ${ECHO} -e "  [ SKIP ]" >> /run/var/bootlog
}

wait_for_user()
{
   # Wait for the user by default
   [ "${HEADLESS=0}" = "0" ] && read ENTER
}

evaluate_retval()
{
   error_value="${?}"

   if [ ${error_value} = 0 ]; then
      echo_ok
   else
      echo_failure
   fi

   # This prevents the 'An Unexpected Error Has Occurred' from trivial
   # errors.
   return 0
}

print_status()
{
   if [ "${#}" = "0" ]; then
      echo "Usage: ${0} {success|warning|failure}"
      return 1
   fi

   case "${1}" in

      success)
         echo_ok
         ;;

      warning)
         # Leave this extra case in because old scripts
         # may call it this way.
         case "${2}" in
            running)
               ${ECHO} -e -n "${CURS_UP}"
               ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
```

```
                boot_mesg "Already running." ${WARNING}
                echo_warning
                ;;
             not_running)
                ${ECHO} -e -n "${CURS_UP}"
                ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                boot_mesg "Not running." ${WARNING}
                echo_warning
                ;;
             not_available)
                ${ECHO} -e -n "${CURS_UP}"
                ${ECHO} -e -n "\\033[${STRING_LENGTH}G   "
                boot_mesg "Not available." ${WARNING}
                echo_warning
                ;;
             *)
                # This is how it is supposed to
                # be called
                echo_warning
                ;;
          esac
       ;;

       failure)
          echo_failure
       ;;

   esac

}

reloadproc()
{
   local pidfile=""
   local failure=0

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" -lt "1" ]; then
      log_failure_msg "Usage: reloadproc [-p pidfile] pathname"
      return 2
   fi
```

```
   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Warn about stale pid file
   if [ "$?" = 1 ]; then
      boot_mesg -n "Removing stale pid file: ${pidfile}. " ${WARNING}
      rm -f "${pidfile}"
   fi

   if [ -n "${pidlist}" ]; then
      for pid in ${pidlist}
      do
         kill -"${RELOADSIG}" "${pid}" || failure="1"
      done

      (exit ${failure})
      evaluate_retval

   else
      boot_mesg "Process ${1} not running." ${WARNING}
      echo_warning
   fi
}

statusproc()
{
   local pidfile=""
   local base=""
   local ret=""

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done
```

```
   if [ "${#}" != "1" ]; then
      shift 1
      log_failure_msg "Usage: statusproc [-p pidfile] pathname"
      return 2
   fi

   # Get the process basename
   base="${1##*/}"

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Store the return status
   ret=$?

   if [ -n "${pidlist}" ]; then
      ${ECHO} -e "${INFO}${base} is running with Process"\
         "ID(s) ${pidlist}.${NORMAL}"
   else
      if [ -n "${base}" -a -e "/var/run/${base}.pid" ]; then
         ${ECHO} -e "${WARNING}${1} is not running but"\
            "/var/run/${base}.pid exists.${NORMAL}"
      else
         if [ -n "${pidfile}" -a -e "${pidfile}" ]; then
            ${ECHO} -e "${WARNING}${1} is not running"\
               "but ${pidfile} exists.${NORMAL}"
         else
            ${ECHO} -e "${INFO}${1} is not running.${NORMAL}"
         fi
      fi
   fi

   # Return the status from pidofproc
   return $ret
}

# The below functions are documented in the LSB-generic 2.1.0

#***********************************************************************
# Function - pidofproc [-s] [-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
#
# Outputs: return 0 - Success, pid's in stdout
#          return 1 - Program is dead, pidfile exists
```

```
#          return 2 - Invalid or excessive number of arguments,
#                     warning in stdout
#          return 3 - Program is not running
#
# Dependencies: pidof, echo, head
#
# Todo: Remove dependency on head
#       This replaces getpids
#       Test changes to pidof
#
#****************************************************************************
pidofproc()
{
   local pidfile=""
   local lpids=""
   local silent=""
   pidlist=""
   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;

         -s)
            # Added for legacy opperation of getpids
            # eliminates several '> /dev/null'
            silent="1"
            shift 1
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" != "1" ]; then
      shift 1
      log_failure_msg "Usage: pidofproc [-s] [-p pidfile] pathname"
      return 2
   fi

   if [ -n "${pidfile}" ]; then
      if [ ! -r "${pidfile}" ]; then
         return 3 # Program is not running
      fi

      lpids=`head -n 1 ${pidfile}`
      for pid in ${lpids}
      do
         if [ "${pid}" -ne "$$" -a "${pid}" -ne "${PPID}" ]; then
            kill -0 "${pid}" 2>/dev/null &&
```

```
            pidlist="${pidlist} ${pid}"
         fi

         if [ "${silent}" != "1" ]; then
            echo "${pidlist}"
         fi

         test -z "${pidlist}" &&
         # Program is dead, pidfile exists
         return 1
         # else
         return 0
      done

   else
      pidlist=`pidof -o $$ -o $PPID -x "$1"`
      if [ "${silent}" != "1" ]; then
         echo "${pidlist}"
      fi

      # Get provide correct running status
      if [ -n "${pidlist}" ]; then
         return 0
      else
         return 3
      fi

   fi

   if [ "$?" != "0" ]; then
      return 3 # Program is not running
   fi
}

#****************************************************************************
# Function - loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]
#
# Purpose: This runs the specified program as a daemon
#
# Inputs: -f, run the program even if it is already running
#         -n nicelevel, specifies a nice level. See nice(1).
#         -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         args, arguments to pass to specified program
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Program or service status is unknown
#
# Dependencies: nice, rm
#
# Todo: LSB says this should be called start_daemon
#       LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#       loadproc returns 0 if program is already running, not LSB compliant
```

```
#
#*****************************************************************************
loadproc()
{
   local pidfile=""
   local forcestart=""
   local nicelevel="10"

# This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

  while true
   do
      case "${1}" in
         -f)
            forcestart="1"
            shift 1
            ;;
         -n)
            nicelevel="${2}"
            shift 2
            ;;
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
            log_failure_msg "Unknown Option: ${1}"
            return 2 #invalid or excess argument(s)
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" = "0" ]; then
      log_failure_msg "Usage: loadproc [-f] [-n nicelevel] [-p pidfile] pathname [args]"
      return 2 #invalid or excess argument(s)
   fi

   if [ -z "${forcestart}" ]; then
      if [ -z "${pidfile}" ]; then
         pidofproc -s "${1}"
      else
         pidofproc -s -p "${pidfile}" "${1}"
      fi

      case "${?}" in
         0)
            log_warning_msg "Unable to continue: ${1} is running"
            return 0 # 4
            ;;
         1)
            boot_mesg "Removing stale pid file: ${pidfile}" ${WARNING}
```

```
            rm -f "${pidfile}"
            ;;
         3)
            ;;
         *)
            log_failure_msg "Unknown error code from pidofproc: ${?}"
            return 4
            ;;
      esac
   fi

   nice -n "${nicelevel}" "${@}"
   evaluate_retval # This is "Probably" not LSB compliant,
#                        but required to be compatible with older bootscripts
   return 0
}

#****************************************************************************
# Function - killproc  [-p pidfile] pathname [signal]
#
# Purpose:
#
# Inputs: -p pidfile, uses the specified pidfile
#         pathname, pathname to the specified program
#         signal, send this signal to pathname
#
# Outputs: return 0 - Success
#          return 2 - Invalid of excessive number of arguments,
#                     warning in stdout
#          return 4 - Unknown Status
#
# Dependencies: kill, rm
#
# Todo: LSB does not say that it should call evaluate_retval
#       It checks for PIDFILE, which is deprecated.
#         Will be removed after BLFS 6.0
#
#****************************************************************************
killproc()
{
   local pidfile=""
   local killsig=TERM # default signal is SIGTERM
   pidlist=""

   # This will ensure compatibility with previous LFS Bootscripts
   if [ -n "${PIDFILE}" ]; then
      pidfile="${PIDFILE}"
   fi

   while true
   do
      case "${1}" in
         -p)
            pidfile="${2}"
            shift 2
            ;;
         -*)
```

```
            log_failure_msg "Unknown Option: ${1}"
            return 2
            ;;
         *)
            break
            ;;
      esac
   done

   if [ "${#}" = "2" ]; then
      killsig="${2}"
   elif [ "${#}" != "1" ]; then
      shift 2
      log_failure_msg "Usage: killproc  [-p pidfile] pathname [signal]"
      return 2
   fi

   # Is the process running?
   if [ -z "${pidfile}" ]; then
      pidofproc -s "${1}"
   else
      pidofproc -s -p "${pidfile}" "${1}"
   fi

   # Remove stale pidfile
   if [ "$?" = 1 ]; then
      boot_mesg "Removing stale pid file: ${pidfile}." ${WARNING}
      rm -f "${pidfile}"
   fi

    # If running, send the signal
    if [ -n "${pidlist}" ]; then
   for pid in ${pidlist}
   do
      kill -${killsig} ${pid} 2>/dev/null

      # Wait up to 3 seconds, for ${pid} to terminate
      case "${killsig}" in
      TERM|SIGTERM|KILL|SIGKILL)
         # sleep in 1/10ths of seconds and
         # multiply KILLDELAY by 10
         local dtime="${KILLDELAY}0"
         while [ "${dtime}" != "0" ]
         do
            kill -0 ${pid} 2>/dev/null || break
            sleep 0.1
            dtime=$(( ${dtime} - 1))
         done
         # If ${pid} is still running, kill it
         kill -0 ${pid} 2>/dev/null && kill -KILL ${pid} 2>/dev/null
         ;;
      esac
   done

   # Check if the process is still running if we tried to stop it
   case "${killsig}" in
   TERM|SIGTERM|KILL|SIGKILL)
```

```
      if [ -z "${pidfile}" ]; then
         pidofproc -s "${1}"
      else
         pidofproc -s -p "${pidfile}" "${1}"
      fi

      # Program was terminated
      if [ "$?" != "0" ]; then
         # Remove the pidfile if necessary
         if [ -f "${pidfile}" ]; then
            rm -f "${pidfile}"
         fi
         echo_ok
         return 0
      else # Program is still running
         echo_failure
         return 4 # Unknown Status
      fi
      ;;
   *)
      # Just see if the kill returned successfully
      evaluate_retval
      ;;
   esac
    else # process not running
   print_status warning not_running
    fi
}


#****************************************************************************
# Function - log_success_msg "message"
#
# Purpose: Print a success message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#****************************************************************************
log_success_msg()
{
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${SUCCESS}""  OK  ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -n -e "${@}  [ OK ]"  >> /run/var/bootlog
   return 0
}

#****************************************************************************
# Function - log_failure_msg "message"
#
```

```
# Purpose: Print a failure message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
log_failure_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${FAILURE}"" FAIL ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ FAIL ]" >> /run/var/bootlog
   return 0
}

#*****************************************************************************
# Function - log_warning_msg "message"
#
# Purpose: print a warning message
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
log_warning_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""["${WARNING}"" WARN ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ WARN ]" >> /run/var/bootlog
   return 0
}

#*****************************************************************************
# Function - log_skipped_msg "message"
#
# Purpose: print a message that the script was skipped
#
# Inputs: $@ - Message
#
# Outputs: Text output to screen
#
# Dependencies: echo
#
# Todo: logging
#
#*****************************************************************************
```

```
log_skipped_msg() {
   ${ECHO} -n -e "${BOOTMESG_PREFIX}${@}"
   ${ECHO} -e "${SET_COL}""${BRACKET}""[""${WARNING}"" SKIP ""${BRACKET}""]""${NORMAL}"

   [ -d /run/var ] || return 0
   ${ECHO} -e "${@}  [ SKIP ]" >> /run/var/bootlog
   return 0
}

# End boot functions
```

D.4. /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:
# 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/var is available before logging any messages
      if ! mountpoint /run >/dev/null; then
         mount /run || failed=1
      fi

      mkdir -p /run/var /run/lock /run/shm
      chmod 1777 /run/shm

      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

      # Copy devices that Udev >= 155 doesn't handle to /dev
      cp -a /lib/udev/devices/* /dev

      ln -sfn /run/shm /dev/shm

      (exit ${failed})
      evaluate_retval
      exit $failed
      ;;

   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End mountvirtfs
```

D.5. /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 sysctl
# 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/ksyms -o -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.6. /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:
# 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

      # Udev handles uevents itself, so we don't need to have
      # the kernel call out to any binary in response to them
      echo > /proc/sys/kernel/hotplug
```

```
      # Start the udev daemon to continually watch for, and act on,
      # uevents
      /lib/udev/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.7. /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:
# 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.8. /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 BLFS
### 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.9. /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 $time
# 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="\n/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.  This 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 [ "${error_value}" -ge 16 ]; then
         msg="\nFAILURE:\n\nUnexpected Failure "
         msg="${msg}running fsck.  Exited with error "
         msg="${msg} code: ${error_value}."
         log_failure_msg $msg
         exit ${error_value}
      fi

      exit 0
      ;;
   *)
      echo "Usage: ${0} {start}"
      exit 1
      ;;
esac

# End checkfs
```

D.10. /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:
# Required-Stop:       swap
# 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 -o remount,rw / >/dev/null
      evaluate_retval

      # Remove fsck-related file system watermarks.
      rm -f /fastboot /forcefsck

      # 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..."
      mount -a -O no_netdev >/dev/null
      evaluate_retval
      exit $failed
      ;;

   stop)
      # Don't unmount virtual file systems like /run
      log_info_msg "Unmounting all other currently mounted file systems..."
      umount -a -d -r -t notmpfs,nosysfs,nodevtmpfs,noproc,nodevpts >/dev/null
      evaluate_retval

      # Make sure / is mounted read only (umount bug)
      mount -o 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.11. /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
# 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.12. /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 /var/run, /var/lock, and
#                      optionally, /tmp.  cleanfs also creates /var/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

      > /var/run/utmp

      if grep -q '^utmp:' /etc/group ; then
         chmod 664 /var/run/utmp
         chgrp utmp /var/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.13. /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:
# Should-Start:        $local_fs
# 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.14. /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:      $local_fs
# Should-Start:
# 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

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.15. /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:
# 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.16. /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:      localnet
# Should-Start:
# Required-Stop:       $local_fs sendsignals
# Should-Stop:
# Default-Start:       2 3 4 5
# Default-Stop:        0 1 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

. /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.17. /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 swap localnet
# Should-Start:        $syslog
# Required-Stop:       $local_fs swap localnet
# Should-Stop:         $syslog
# 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)
      # 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

         /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.18. /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 "${1}" in
   stop)
      log_info_msg "Sending all processes the TERM signal..."
      killall5 -15
      error_value=${?}

      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
      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.19. /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.20. /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.21. /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.22. /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 it's 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.23. /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.24. /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.25. /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
#
# Version     : LFS 7.2
#
# 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()
{
  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 "\nInterface ${IFACE} doesn't exist."
     exit 1
  fi
}

RELEASE="7.2"

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

log_info_msg "Bringing up the ${1} interface... "

if [ ! -r "${file}" ]; then
   log_failure_msg2 "${file} is missing or cannot be accessed."
   exit 1
fi

. $file

if [ "$IFACE" = "" ]; then
   log_failure_msg2 "${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
   log_info_msg2 "skipped"
   exit 0
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

# Create/configure the interface
```

```
for S in ${SERVICE}; do
  IFCONFIG=${file} /lib/services/${S} ${IFACE} up
done

# Bring up the interface and any components
for I in $IFACE $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 "\nGateway already setup; skipping."
   else
      log_info_msg "Setting up default gateway..."
      ip route add default via ${GATEWAY} dev ${IFACE}
      evaluate_retval
   fi
fi

# End /sbin/ifup
```

D.26. /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.27. /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 "${BROADCAST}" ]; then
   args="${args} broadcast ${BROADCAST}"
fi

case "${2}" in
   up)
      if [ "$(ip addr show ${1} 2>/dev/null | grep ${IP}/)" = "" ]; then

         # Cosmetic output not needed for multiple services
         if ! $(echo ${SERVICE} | grep -q " "); then
           log_info_msg2 "\n" # Terminate the previous message
         fi

         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.28. /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 do pacote udev-lfs-208-3.tar.bz2 neste apêndice são listadas por conveniência. Instalação é normalmente feito via instruções em Seção 6.60, “Udev-208 (Extraído de systemd-208)”.

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 está licenciado sob a licença Creative Commons Attribution-NonCommercial-ShareAlike 2.0.

Instruções de computador podem ser extraídas deste livros sob os termos da licença MIT.

F.1. Creative Commons License

Creative Commons Legal Code

Attribution-NonCommercial-ShareAlike 2.0

Importante

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM ITS USE.

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions
   a. "Collective Work" means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.
   b. "Derivative Work" means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.
   c. "Licensor" means the individual or entity that offers the Work under the terms of this License.
   d. "Original Author" means the individual or entity who created the Work.
   e. "Work" means the copyrightable work of authorship offered under the terms of this License.
   f. "You" means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

g. "License Elements" means the following high-level license attributes as selected by Licensor and indicated in the title of this License: Attribution, Noncommercial, ShareAlike.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

   a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

   b. to create and reproduce Derivative Works;

   c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

   d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

   The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Sections 4(e) and 4(f).

4. Restrictions.The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

   a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any reference to such Licensor or the Original Author, as requested.

   b. You may distribute, publicly display, publicly perform, or publicly digitally perform a Derivative Work only under the terms of this License, a later version of this License with the same License Elements as this License, or a Creative Commons iCommons license that contains the same License Elements as this License (e.g. Attribution-NonCommercial-ShareAlike 2.0 Japan). You must include a copy of, or the Uniform Resource Identifier for, this License or other license specified in the previous sentence with every copy or phonorecord of each Derivative Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Derivative Works that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder, and You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Derivative Work with any technological measures that control access or use of the Work in a manner

inconsistent with the terms of this License Agreement. The above applies to the Derivative Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Derivative Work itself to be made subject to the terms of this License.

c. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

d. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

e. For the avoidance of doubt, where the Work is a musical composition:

   i. Performance Royalties Under Blanket Licenses. Licensor reserves the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work if that performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

   ii. Mechanical Rights and Statutory Royalties. Licensor reserves the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions), if Your distribution of such cover version is primarily intended for or directed toward commercial advantage or private monetary compensation. 6. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

f. Webcasting Rights and Statutory Royalties. For the avoidance of doubt, where the Work is a sound recording, Licensor reserves the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions), if Your public digital performance is primarily intended for or directed toward commercial advantage or private monetary compensation.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

   b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

   a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

   b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

   c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

   e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

Importante

Creative Commons is not a party to this License, and makes no warranty whatsoever in connection with the Work. Creative Commons will not be liable to You or any party on any legal theory for any damages whatsoever, including without limitation any general, special, incidental or consequential damages arising in connection to this license. Notwithstanding the foregoing two (2) sentences, if Creative Commons has expressly identified itself as the Licensor hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the Work is licensed under the CCPL, neither party will use the trademark "Creative Commons" or any related trademark or logo of Creative Commons without the prior written consent of Creative Commons. Any permitted use will be in compliance with Creative Commons' then-current trademark usage guidelines, as may be published on its website or otherwise made available upon request from time to time.

Creative Commons may be contacted at *http://creativecommons.org/*.

F.2. The MIT License

Copyright © 1999-2014 Gerard Beekmans

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Índice Remissivo

Pacotes

Autoconf: 154
Automake: 156
Bash: 144
  tools: 59
Bash: 144
  tools: 59
Bc: 146
Binutils: 103
  tools, pass 1: 35
  tools, pass 2: 46
Binutils: 103
  tools, pass 1: 35
  tools, pass 2: 46
Binutils: 103
  tools, pass 1: 35
  tools, pass 2: 46
Bison: 140
Bootscripts - Scripts de Inicialização: 214
  uso: 216
Bootscripts - Scripts de Inicialização: 214
  uso: 216
Bzip2: 116
  tools: 60
Bzip2: 116
  tools: 60
Check: 57
Coreutils: 131
  tools: 61
Coreutils: 131
  tools: 61
DejaGNU: 56
Diffutils: 158
  tools: 62
Diffutils: 158
  tools: 62
E2fsprogs: 128
Expect: 54
File: 102
  tools: 63
File: 102
  tools: 63
Findutils: 160
  tools: 64
Findutils: 160
  tools: 64
Flex: 138
Gawk: 159
  tools: 65
Gawk: 159
  tools: 65
GCC: 110
  tools, libstdc++: 44
  tools, pass 1: 37
  tools, pass 2: 48
GCC: 110
  tools, libstdc++: 44
  tools, pass 1: 37
  tools, pass 2: 48
GCC: 110
  tools, libstdc++: 44
  tools, pass 1: 37
  tools, pass 2: 48
GCC: 110
  tools, libstdc++: 44
  tools, pass 1: 37
  tools, pass 2: 48
GDBM: 148
Gettext: 162
  tools: 66
Gettext: 162
  tools: 66
Glibc: 92
  tools: 41
Glibc: 92
  tools: 41
GMP: 106
Grep: 141
  tools: 67
Grep: 141
  tools: 67
Groff: 164
GRUB: 169
Gzip: 172
  tools: 68
Gzip: 172
  tools: 68
Iana-Etc: 136
Inetutils: 149
IPRoute2: 174
Kbd: 176

Kmod: 178
Less: 171
Libpipeline: 180
Libtool: 147
Linux: 231
  API headers: 90
  tools, API headers: 40
Linux: 231
  API headers: 90
  tools, API headers: 40
Linux: 231
  API headers: 90
  tools, API headers: 40
M4: 137
  tools: 69
M4: 137
  tools: 69
Make: 181
  tools: 70
Make: 181
  tools: 70
Man-DB: 195
Man-pages: 91
MPC: 109
MPFR: 108
Ncurses: 119
  tools: 58
Ncurses: 119
  tools: 58
Patch: 182
  tools: 71
Patch: 182
  tools: 71
Perl: 151
  tools: 72
Perl: 151
  tools: 72
Pkgconfig: 118
Procps-ng: 126
Psmisc: 125
rc.site: 223
Readline: 142
Sed: 115
  tools: 73
Sed: 115
  tools: 73
Shadow: 122
  configuring: 123
Shadow: 122
  configuring: 123
Sysklogd: 183
  configuring: 183
Sysklogd: 183
  configuring: 183
Sysvinit: 184
  configuring: 217
Sysvinit: 184
  configuring: 217
Tar: 185
  tools: 74
Tar: 185
  tools: 74
Tcl: 52
Texinfo: 186
  tools: 75
Texinfo: 186
  tools: 75
Udev: 188
  uso: 207
Udev: 188
  uso: 207
Util-linux: 76, 190
Util-linux: 76, 190
Vim: 198
Xz: 167
  tools: 77
Xz: 167
  tools: 77
Zlib: 101

Programas

a2p: 151, 152
accessdb: 195, 196
acinstall: 156, 156
aclocal: 156, 156
aclocal-1.14: 156, 156
addftinfo: 164, 164
addpart: 190, 191
addr2line: 103, 104
afmtodit: 164, 164
agetty: 190, 191
apropos: 195, 197
ar: 103, 104
as: 103, 104

ata_id: 188, 189
autoconf: 154, 154
autoheader: 154, 154
autom4te: 154, 154
automake: 156, 156
automake-1.14: 156, 156
autopoint: 162, 162
autoreconf: 154, 154
autoscan: 154, 154
autoupdate: 154, 154
awk: 159, 159
badblocks: 128, 129
base64: 131, 132
basename: 131, 132
bash: 144, 145
bashbug: 144, 145
bc: 146, 146
bigram: 160, 160
bison: 140, 140
blkdiscard: 190, 191
blkid: 190, 191
blockdev: 190, 191
bootlogd: 184, 184
bridge: 174, 174
bunzip2: 116, 117
bzcat: 116, 117
bzcmp: 116, 117
bzdiff: 116, 117
bzegrep: 116, 117
bzfgrep: 116, 117
bzgrep: 116, 117
bzip2: 116, 117
bzip2recover: 116, 117
bzless: 116, 117
bzmore: 116, 117
c++: 110, 113
c++filt: 103, 104
c2ph: 151, 152
cal: 190, 191
captoinfo: 119, 121
cat: 131, 132
catchsegv: 92, 97
catman: 195, 197
cc: 110, 113
cdrom_id: 188, 189
cfdisk: 190, 191
chage: 122, 124
chattr: 128, 129
chcon: 131, 132
chcpu: 190, 191
checkmk: 57, 57
chem: 164, 164
chfn: 122, 124
chgpasswd: 122, 124
chgrp: 131, 132
chmod: 131, 132
chown: 131, 132
chpasswd: 122, 124
chroot: 131, 132
chrt: 190, 191
chsh: 122, 124
chvt: 176, 177
cksum: 131, 132
clear: 119, 121
cmp: 158, 158
code: 160, 160
col: 190, 191
colcrt: 190, 191
collect: 188, 189
colrm: 190, 191
column: 190, 191
comm: 131, 133
compile: 156, 156
compile_et: 128, 129
config.charset: 162, 162
config.guess: 156, 156
config.rpath: 162, 162
config.sub: 156, 156
config_data: 151, 152
corelist: 151, 152
cp: 131, 133
cpan: 151, 152
cpan2dist: 151, 152
cpanp: 151, 152
cpanp-run-perl: 151, 152
cpp: 110, 113
csplit: 131, 133
ctrlaltdel: 190, 191
ctstat: 174, 174
cut: 131, 133
cytune: 190, 191
date: 131, 133
dc: 146, 146
dd: 131, 133

deallocvt: 176, 177
debugfs: 128, 130
delpart: 190, 191
depcomp: 156, 156
depmod: 178, 179
df: 131, 133
diff: 158, 158
diff3: 158, 158
dir: 131, 133
dircolors: 131, 133
dirname: 131, 133
dmesg: 190, 191
du: 131, 133
dumpe2fs: 128, 130
dumpkeys: 176, 177
e2freefrag: 128, 130
e2fsck: 128, 130
e2image: 128, 130
e2label: 128, 130
e2undo: 128, 130
e4defrag: 128, 130
echo: 131, 133
egrep: 141, 141
eject: 190, 191
elfedit: 103, 104
enc2xs: 151, 152
env: 131, 133
envsubst: 162, 162
eqn: 164, 164
eqn2graph: 164, 164
ex: 198, 200
expand: 131, 133
expect: 54, 55
expiry: 122, 124
expr: 131, 133
factor: 131, 133
faillog: 122, 124
fallocate: 190, 191
false: 131, 133
fdformat: 190, 191
fdisk: 190, 191
fgconsole: 176, 177
fgrep: 141, 141
arquivo: 102, 102
filefrag: 128, 130
find: 160, 160
find2perl: 151, 152
findfs: 190, 191
findmnt: 190, 192
flex: 138, 138
flex++: 138, 139
flock: 190, 192
fmt: 131, 133
fold: 131, 133
frcode: 160, 160
free: 126, 126
fsck: 190, 192
fsck.cramfs: 190, 192
fsck.ext2: 128, 130
fsck.ext3: 128, 130
fsck.ext4: 128, 130
fsck.ext4dev: 128, 130
fsck.minix: 190, 192
fsfreeze: 190, 192
fstab-decode: 184, 184
fstrim: 190, 192
ftp: 149, 150
fuser: 125, 125
g++: 110, 113
gawk: 159, 159
gawk-4.1.0: 159, 159
gcc: 110, 113
gc-ar: 110, 113
gc-nm: 110, 114
gc-ranlib: 110, 114
gcov: 110, 114
gdiffmk: 164, 165
gencat: 92, 97
genl: 174, 174
geqn: 164, 165
getconf: 92, 97
getent: 92, 97
getkeycodes: 176, 177
getopt: 190, 192
gettext: 162, 162
gettext.sh: 162, 162
gettextize: 162, 162
gpasswd: 122, 124
gprof: 103, 104
grap2graph: 164, 165
grep: 141, 141
grn: 164, 165
grodvi: 164, 165
groff: 164, 165

groffer: 164, 165
grog: 164, 165
grolbp: 164, 165
grolj4: 164, 165
grops: 164, 165
grotty: 164, 165
groupadd: 122, 124
groupdel: 122, 124
groupmems: 122, 124
groupmod: 122, 124
groups: 131, 133
grpck: 122, 124
grpconv: 122, 124
grpunconv: 122, 124
grub-bios-setup: 169, 169
grub-editenv: 169, 169
grub-fstest: 169, 169
grub-install: 169, 170
grub-kbdcomp: 169, 170
grub-menulst2cfg: 169, 170
grub-mkconfig: 169, 170
grub-mkimage: 169, 170
grub-mklayout: 169, 170
grub-mknetdir: 169, 170
grub-mkpasswd-pbkdf2: 169, 170
grub-mkrelpath: 169, 170
grub-mkrescue: 169, 170
grub-mkstandalone: 169, 170
grub-ofpathname: 169, 170
grub-probe: 169, 170
grub-reboot: 169, 170
grub-script-check: 169, 170
grub-set-default: 169, 170
grub-setup: 169, 170
gtbl: 164, 165
gunzip: 172, 172
gzexe: 172, 172
gzip: 172, 172
h2ph: 151, 152
h2xs: 151, 152
halt: 184, 184
head: 131, 133
hexdump: 190, 192
hostid: 131, 133
hostname: 149, 150
hostname: 162, 162
hpftodit: 164, 165
hwclock: 190, 192
i386: 190, 192
iconv: 92, 97
iconvconfig: 92, 97
id: 131, 133
ifcfg: 174, 175
ifconfig: 149, 150
ifnames: 154, 154
ifstat: 174, 175
igawk: 159, 159
indxbib: 164, 165
info: 186, 187
infocmp: 119, 121
infokey: 186, 187
infotocap: 119, 121
init: 184, 184
insmod: 178, 179
install: 131, 133
install-info: 186, 187
install-sh: 156, 157
instmodsh: 151, 152
ionice: 190, 192
ip: 174, 175
ipcmk: 190, 192
ipcrm: 190, 192
ipcs: 190, 192
isosize: 190, 192
join: 131, 133
json_pp: 151, 152
kbdinfo: 176, 177
kbdrate: 176, 177
kbd_mode: 176, 177
kill: 190, 192
killall: 125, 125
killall5: 184, 184
klogd: 183, 183
kmod: 178, 179
last: 190, 192
lastb: 190, 192
lastlog: 122, 124
ld: 103, 104
ld.bfd: 103, 104
ldattach: 190, 192
ldconfig: 92, 97
ldd: 92, 97
lddlibc4: 92, 97
less: 171, 171

lessecho: 171, 171
lesskey: 171, 171
lex: 138, 139
lexgrog: 195, 197
lfskernel-3.13.3: 231, 233
libasan: 110, 114
libnetcfg: 151, 152
libtool: 147, 147
libtoolize: 147, 147
link: 131, 133
linux32: 190, 192
linux64: 190, 192
lkbib: 164, 165
ln: 131, 133
lnstat: 174, 175
loadkeys: 176, 177
loadunimap: 176, 177
locale: 92, 97
localedef: 92, 97
locate: 160, 160
logger: 190, 192
login: 122, 124
logname: 131, 133
logoutd: 122, 124
logsave: 128, 130
look: 190, 192
lookbib: 164, 165
losetup: 190, 192
ls: 131, 133
lsattr: 128, 130
lsblk: 190, 192
lscpu: 190, 192
lslocks: 190, 192
lsmod: 178, 179
lzcat: 167, 167
lzcmp: 167, 167
lzdiff: 167, 167
lzegrep: 167, 167
lzfgrep: 167, 167
lzgrep: 167, 167
lzless: 167, 167
lzma: 167, 167
lzmadec: 167, 168
lzmainfo: 167, 168
lzmore: 167, 168
m4: 137, 137
make: 181, 181
makedb: 92, 97
makeinfo: 186, 187
man: 195, 197
mandb: 195, 197
manpath: 195, 197
mapscrn: 176, 177
mcookie: 190, 192
md5sum: 131, 133
mdate-sh: 156, 157
mesg: 190, 192
missing: 156, 157
mkdir: 131, 133
mke2fs: 128, 130
mkfifo: 131, 134
mkfs: 190, 192
mkfs.bfs: 190, 192
mkfs.cramfs: 190, 192
mkfs.ext2: 128, 130
mkfs.ext3: 128, 130
mkfs.ext4: 128, 130
mkfs.ext4dev: 128, 130
mkfs.minix: 190, 192
mkinstalldirs: 156, 157
mklost+found: 128, 130
mknod: 131, 134
mkswap: 190, 193
mktemp: 131, 134
mk_cmds: 128, 130
mmroff: 164, 165
modinfo: 178, 179
modprobe: 178, 179
more: 190, 193
mount: 190, 193
mountpoint: 190, 193
msgattrib: 162, 163
msgcat: 162, 163
msgcmp: 162, 163
msgcomm: 162, 163
msgconv: 162, 163
msgen: 162, 163
msgexec: 162, 163
msgfilter: 162, 163
msgfmt: 162, 163
msggrep: 162, 163
msginit: 162, 163
msgmerge: 162, 163
msgunfmt: 162, 163

msguniq: 162, 163
mtrace: 92, 97
mv: 131, 134
namei: 190, 193
ncursesw5-config: 119, 121
neqn: 164, 165
newgrp: 122, 124
newusers: 122, 124
ngettext: 162, 163
nice: 131, 134
nl: 131, 134
nm: 103, 104
nohup: 131, 134
nologin: 122, 124
nproc: 131, 134
nroff: 164, 165
nscd: 92, 97
nsenter: 190, 193
nstat: 174, 175
numfmt: 131, 134
objcopy: 103, 104
objdump: 103, 104
od: 131, 134
oldfind: 160, 161
openvt: 176, 177
partx: 190, 193
passwd: 122, 124
paste: 131, 134
patch: 182, 182
pathchk: 131, 134
pcprofiledump: 92, 97
pdfroff: 164, 165
pdftexi2dvi: 186, 187
peekfd: 125, 125
perl: 151, 152
perl5.18.2: 151, 152
perlbug: 151, 152
perldoc: 151, 152
perlivp: 151, 152
perlthanks: 151, 153
pfbtops: 164, 165
pg: 190, 193
pgrep: 126, 126
pic: 164, 165
pic2graph: 164, 165
piconv: 151, 153
pidof: 126, 127
ping: 149, 150
ping6: 149, 150
pinky: 131, 134
pivot_root: 190, 193
pkg-config: 118, 118
pkill: 126, 127
pl2pm: 151, 153
pldd: 92, 97
pmap: 126, 127
pod2html: 151, 153
pod2latex: 151, 153
pod2man: 151, 153
pod2texi: 186, 187
pod2text: 151, 153
pod2usage: 151, 153
podchecker: 151, 153
podselect: 151, 153
post-grohtml: 164, 165
poweroff: 184, 184
pr: 131, 134
pre-grohtml: 164, 165
preconv: 164, 165
printenv: 131, 134
printf: 131, 134
prlimit: 190, 193
prove: 151, 153
prtstat: 125, 125
ps: 126, 127
psed: 151, 153
psfaddtable: 176, 177
psfgettable: 176, 177
psfstriptable: 176, 177
psfxtable: 176, 177
pstree: 125, 125
pstree.x11: 125, 125
pstruct: 151, 153
ptar: 151, 153
ptardiff: 151, 153
ptargrep: 151, 153
ptx: 131, 134
pwck: 122, 124
pwconv: 122, 124
pwd: 131, 134
pwdx: 126, 127
pwunconv: 122, 124
py-compile: 156, 157
ranlib: 103, 104

raw: 190, 193
rcp: 149, 150
readelf: 103, 104
readlink: 131, 134
readprofile: 190, 193
realpath: 131, 134
reboot: 184, 184
recode-sr-latin: 162, 163
refer: 164, 166
rename: 190, 193
renice: 190, 193
reset: 119, 121
resize2fs: 128, 130
resizepart: 190, 193
rev: 190, 193
rexec: 149, 150
rlogin: 149, 150
rm: 131, 134
rmdir: 131, 134
rmmod: 178, 179
rmt: 185, 185
roff2dvi: 164, 166
roff2html: 164, 166
roff2pdf: 164, 166
roff2ps: 164, 166
roff2text: 164, 166
roff2x: 164, 166
routef: 174, 175
routel: 174, 175
rpcgen: 92, 97
rsh: 149, 150
rtacct: 174, 175
rtcwake: 190, 193
rtmon: 174, 175
rtpr: 174, 175
rtstat: 174, 175
runcon: 131, 134
runlevel: 184, 184
runtest: 56, 56
rview: 198, 200
rvim: 198, 200
s2p: 151, 153
script: 190, 193
scriptreplay: 190, 193
scsi_id: 188, 189
sdiff: 158, 158
sed: 115, 115
seq: 131, 134
setarch: 190, 193
setfont: 176, 177
setkeycodes: 176, 177
setleds: 176, 177
setmetamode: 176, 177
setsid: 190, 193
setterm: 190, 193
sfdisk: 190, 193
sg: 122, 124
sh: 144, 145
sha1sum: 131, 134
sha224sum: 131, 134
sha256sum: 131, 134
sha384sum: 131, 134
sha512sum: 131, 134
shasum: 151, 153
showconsolefont: 176, 177
showkey: 176, 177
shred: 131, 134
shuf: 131, 134
shutdown: 184, 184
size: 103, 104
slabtop: 126, 127
sleep: 131, 134
sln: 92, 97
soelim: 164, 166
sort: 131, 135
sotruss: 92, 97
splain: 151, 153
split: 131, 135
sprof: 92, 98
ss: 174, 175
stat: 131, 135
stdbuf: 131, 135
strings: 103, 104
strip: 103, 105
stty: 131, 135
su: 122, 124
sulogin: 190, 193
sum: 131, 135
swaplabel: 190, 193
swapoff: 190, 193
swapon: 190, 193
switch_root: 190, 193
sync: 131, 135
sysctl: 126, 127

syslogd: 183, 183
tabs: 119, 121
tac: 131, 135
tail: 131, 135
tailf: 190, 193
talk: 149, 150
tar: 185, 185
taskset: 190, 193
tbl: 164, 166
tc: 174, 175
tclsh: 52, 53
tclsh8.6: 52, 53
tee: 131, 135
telinit: 184, 184
telnet: 149, 150
test: 131, 135
testgdbm: 148, 148
texi2dvi: 186, 187
texi2pdf: 186, 187
texi2any: 186, 187
texindex: 186, 187
tfmtodit: 164, 166
tftp: 149, 150
tic: 119, 121
timeout: 131, 135
tload: 126, 127
toe: 119, 121
top: 126, 127
touch: 131, 135
tput: 119, 121
tr: 131, 135
traceroute: 149, 150
troff: 164, 166
true: 131, 135
truncate: 131, 135
tset: 119, 121
tsort: 131, 135
tty: 131, 135
tune2fs: 128, 130
tzselect: 92, 98
udevadm: 188, 189
udevd: 188, 189
ul: 190, 193
umount: 190, 193
uname: 131, 135
uncompress: 172, 172
unexpand: 131, 135
unicode_start: 176, 177
unicode_stop: 176, 177
uniq: 131, 135
unlink: 131, 135
unlzma: 167, 168
unshare: 190, 194
unxz: 167, 168
updatedb: 160, 161
uptime: 126, 127
useradd: 122, 124
userdel: 122, 124
usermod: 122, 124
users: 131, 135
utmpdump: 190, 194
uuidd: 190, 194
uuidgen: 190, 194
vdir: 131, 135
vi: 198, 200
view: 198, 200
vigr: 122, 124
vim: 198, 200
vimdiff: 198, 200
vimtutor: 198, 200
vipw: 122, 124
vmstat: 126, 127
w: 126, 127
wall: 190, 194
watch: 126, 127
wc: 131, 135
wdctl: 190, 194
whatis: 195, 197
whereis: 190, 194
who: 131, 135
whoami: 131, 135
wipefs: 190, 194
x86_64: 190, 194
xargs: 160, 161
xgettext: 162, 163
xsubpp: 151, 153
xtrace: 92, 98
xxd: 198, 200
xz: 167, 168
xzcat: 167, 168
xzcmp: 167, 168
xzdec: 167, 168
xzdiff: 167, 168
xzegrep: 167, 168

xzfgrep: 167, 168
xzgrep: 167, 168
xzless: 167, 168
xzmore: 167, 168
yacc: 140, 140
yes: 131, 135
ylwrap: 156, 157
zcat: 172, 172
zcmp: 172, 172
zdiff: 172, 172
zdump: 92, 98
zegrep: 172, 172
zfgrep: 172, 172
zforce: 172, 172
zgrep: 172, 173
zic: 92, 98
zipdetails: 151, 153
zless: 172, 173
zmore: 172, 173
znew: 172, 173
zsoelim: 195, 197

Bibliotecas

ld.so: 92, 98
libanl: 92, 98
libasprintf: 162, 163
libbfd: 103, 105
libblkid: 190, 194
libBrokenLocale: 92, 98
libbz2*: 116, 117
libc: 92, 98
libcheck: 57, 57
libcidn: 92, 98
libcom_err: 128, 130
libcrypt: 92, 98
libcurses: 119, 121
libdl: 92, 98
libe2p: 128, 130
libexpect-5.45: 54, 55
libext2fs: 128, 130
libfl: 138, 139
libform: 119, 121
libg: 92, 98
libgcc*: 110, 114
libgcov: 110, 114
libgdbm: 148, 148
libgettextlib: 162, 163
libgettextpo: 162, 163
libgettextsrc: 162, 163
libgmp: 106, 107
libgmpxx: 106, 107
libgomp: 110, 114
libhistory: 142, 143
libiberty: 110, 114
libieee: 92, 98
libkmod: 178
libltdl: 147, 147
liblto_plugin*: 110, 114
liblzma*: 167, 168
libm: 92, 98
libmagic: 102, 102
libman: 195, 197
libmandb: 195, 197
libmcheck: 92, 98
libmemusage: 92, 98
libmenu: 119, 121
libmount: 190, 194
libmpc: 109, 109
libmpfr: 108, 108
libmudflap*: 110, 114
libncurses: 119, 121
libnsl: 92, 98
libnss: 92, 98
libopcodes: 103, 105
libpanel: 119, 121
libpcprofile: 92, 98
libpipeline: 180
libprocps: 126, 127
libpthread: 92, 98
libquadmath*: 110, 114
libquota: 128, 130
libreadline: 142, 143
libresolv: 92, 98
librpcsvc: 92, 98
librt: 92, 98
libSegFault: 92, 98
libss: 128, 130
libssp*: 110, 114
libstdbuf.so: 131, 135
libstdc++: 110, 114
libsupc++: 110, 114
libtcl8.6.so: 52, 53
libtclstub8.6.a: 52, 53
libthread_db: 92, 98

libtsan: 110, 114
libudev: 188, 189
libutil: 92, 98
libuuid: 190, 194
liby.a: 140, 140
libz: 101, 101
preloadable_libintl: 162, 163

Scripts

checkfs: 214, 214
cleanfs: 214, 214
console: 214, 214
  configuring: 219
console: 214, 214
  configuring: 219
functions: 214, 214
halt: 214, 214
hostname
  configuring: 218
ifdown: 214, 214
ifup: 214, 214
ipv4-static: 214, 215
localnet: 214, 214
  /etc/hosts: 206
localnet: 214, 214
  /etc/hosts: 206
modules: 214, 214
mountfs: 214, 214
mountvirtfs: 214, 214
network: 214, 214
  /etc/hosts: 206
  configuring: 203
network: 214, 214
  /etc/hosts: 206
  configuring: 203
network: 214, 214
  /etc/hosts: 206
  configuring: 203
rc: 214, 214
reboot: 214, 214
sendsignals: 214, 215
setclock: 214, 215
  configuring: 219
setclock: 214, 215
  configuring: 219
swap: 214, 215
sysctl: 214, 215
sysklogd: 214, 215
  configuring: 222
sysklogd: 214, 215
  configuring: 222
template: 214, 215
udev: 214, 215
udev_retry: 214, 215

outros

/boot/config-3.13.3: 231, 233
/boot/System.map-3.13.3: 231, 233
/dev/*: 80
/etc/fstab: 229
/etc/group: 87
/etc/hosts: 206
/etc/inittab: 217
/etc/inputrc: 227
/etc/ld.so.conf: 96
/etc/lfs-release: 236
/etc/localtime: 94
/etc/modprobe.d/usb.conf: 233
/etc/nsswitch.conf: 94
/etc/passwd: 87
/etc/profile: 225
/etc/protocols: 136
/etc/resolv.conf: 206
/etc/services: 136
/etc/syslog.conf: 183
/etc/udev: 188, 189
/etc/vimrc: 199
/usr/include/asm-generic/*.h: 90, 90
/usr/include/asm/*.h: 90, 90
/usr/include/drm/*.h: 90, 90
/usr/include/linux/*.h: 90, 90
/usr/include/mtd/*.h: 90, 90
/usr/include/rdma/*.h: 90, 90
/usr/include/scsi/*.h: 90, 90
/usr/include/sound/*.h: 90, 90
/usr/include/video/*.h: 90, 90
/usr/include/xen/*.h: 90, 90
/var/log/btmp: 87
/var/log/lastlog: 87
/var/log/wtmp: 87
/var/run/utmp: 87
man pages: 91, 91