

UNIVERSIDADE ESTADUAL DO PARANÁ – UEM
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE SISTEMAS PARA
WEB

EDGAR FEITOSA NORTE

LINUX PERSONALIZADO
BASEADO NO PROJETO LINUX FROM SCRATCH

MARINGÁ

2010

EDGAR FEITOSA NORTE

LINUX PERSONALIZADO

BASEADO NO PROJETO LINUX FROM SCRATCH

Trabalho de conclusão de curso apresentado como requisito para a obtenção de título de especialista em Desenvolvimento de Sistemas para Web da Universidade Estadual do Paraná.

Orientador: *Prof. MSc. Osvaldo Alves dos Santos*

MARINGÁ

2010

Linux From Scratch: Versão 6.3

by Gerard Beekmans

Copyright © 1999–2007 Gerard Beekmans

Copyright (c) 1999–2007, Gerard Beekmans

Todos os direitos reservados.

É permitida o uso e a redistribuição dos fontes e binários, com ou sem modificações, contanto que as seguintes considerações seja atendidas:

- Redistribuições sob qualquer formato não podem ocultar a nota sobre copyright acima, esta lista de considerações e os esclarecimentos que se seguem
- Tanto o nome “Linux From Scratch” quanto os nomes das pessoas que contribuíram podem ser usados para endossar ou promover produtos derivados deste matéria sem permissão prévia, específica e por escrito
- Qualquer materia derivado do Linux From Scratch deve conter uma referência para o projeto “Linux From Scratch”

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RESUMO

Este trabalho demonstra num passo-a-passo resumido baseado no Projeto LFS (Linux From Scratch) como criar um sistema Linux com um mínimo necessário de programas instalados para seu perfeito funcionamento. A intenção é que se entenda como funciona a instalação de um programa no Linux a partir de sua fonte, aprendendo a compilar e instalar o programa de forma manual. Além disso, o projeto estimula o aprendizado e o conhecimento de como configurar um programa e conhecer sua utilidade no sistema, controlando melhor o que precisa ou não ser instalado, economizando com isso espaço em disco e evitando consumo desnecessário da CPU por algum programa instalado e que não estivesse efetivamente em uso. Mas para chegar a este resultado o caminho é árduo, pois tudo é realizado manualmente, sem pacotes prontos que instalam com um simples comando. Deste modo, para que os procedimentos sejam compreendidos e executados com precisão faz-se necessário deter algum conhecimento de Linux. Ao final teremos um sistema mínimo sem muitos recursos, mas totalmente funcional, e com o conhecimento adquirido é perfeitamente possível ampliar seu sistema com novos recursos de acordo com a necessidade.

ABSTRACT

This work demonstrates a step-by-step summary based on the LFS Project (Linux From Scratch) to create a Linux system with minimum required programs installed for flawless performance. The intention is to understand how the installation of a program works on Linux from its source, learning to build and install the program manually. In addition, the project encourages the learning and the knowledge of how to set up a program and how to know its usefulness in the system, having better control on what is needed to be installed or not, thereby saving space on the disk and avoiding unnecessary consumption of CPU by some installed program that was not actually in use. To achieve this result, the path is hard, because everything is done manually, with no ready made packages that install with a simple command. Thus, for which the procedures are understood and executed with precision it is necessary to have some knowledge on Linux. At the end we will have a minimal system without many resources, but fully functional, and with the knowledge obtained it is perfectly possible to expand the system with new features as needed.

SUMÁRIO

1. INTRODUÇÃO	13
2. PRÉ-REQUISITOS	17
3. REQUISITOS DO SISTEMA ANFITRIÃO.....	17
4. CONVENÇÕES UTILIZADAS NESTE DOCUMENTO	18
5. ESTRUTURA.....	19
5.1. Parte I – INTRODUÇÃO.....	19
5.2. Parte II – PREPARANDO A CONFIGURAÇÃO	19
5.3. Parte III – CONSTRUINDO O SISTEMA LFS.....	20
PARTE I - INTRODUÇÃO.....	21
CAPÍTULO 1. INTRODUÇÃO	22
PARTE II – PREPARANDO A CONFIGURAÇÃO	24
CAPÍTULO 2. CARREGANDO O SISTEMA ANFITRIÃO	25
2.2. CRIANDO UMA NOVA PARTIÇÃO.....	28
2.3. CRIANDO O SISTEMA DE ARQUIVOS	29
CAPÍTULO 3. PACOTES E PATCHES	31
3.1. INTRODUÇÃO	31
CAPÍTULO 4. PREPARAÇÕES FINAIS.....	32
4.1. SOBRE A VARIÁVEL \$LFS.....	32
4.2. CRIANDO O DIRETÓRIO \$LFS/tools.....	33
4.3. CRIANDO O USUÁRIO E GRUPO DO SISTEMA LFS.....	33
4.4. CONFIGURANDO O AMBIENTE.....	35
CAPÍTULO 5. CRIANDO UM SISTEMA PROVISÓRIO.....	37
5.1. INTRODUÇÃO	37
5.2. INSTALANDO PACOTES PROVISÓRIOS	37
5.2. Binutils-2.17 (Passo 1).....	37
5.3. GCC-4.1.2 (Passo 1).....	39
5.4. Linux-Libc-Headers-2.6.22.5 API HEADERS.....	41
5.5. Glibc-2.5.1.....	41
5.6. AJUSTANDO AS FERRAMENTAS PROVISÓRIAS.....	44

5.7. Tcl-8.4.15	46
5.8. Expect-5.43.0	47
5.9. DejaGNU-1.4.4.....	49
5.10. GCC-4.1.2 (Passo 2).....	50
5.11. Binutils-2.17 (Passo 2).....	54
5.12. NCurses-5.6	56
5.13. Bash-3.2.....	57
5.14. Bzip2-1.0.4.....	58
5.15. Coreutils-6.9	59
5.16. Diffutils-2.8.1	60
5.17. Findutils-4.2.31	61
5.18. Gawk-3.1.5	62
5.19. Gettext-0.16.1	63
5.20. Grep-2.5.1a	64
5.21. Gzip-1.3.12	65
5.22. Make-3.81	66
5.23. Patch-2.5.4	67
5.24. Perl-5.8.8	67
5.25. Sed-4.1.5	69
5.26. Tar-1.18.....	69
5.27. Texinfo-4.9.....	70
5.28. Util-linux-2.12r	71
5.29. Alternado propriedade	72
PARTE III – CONSTRUINDO O SISTEMA LFS.....	74
CAPÍTULO 6 – INSTALANDO O SOFTWARE DO SISTEMA BÁSICO.....	75
6.1. INTRODUÇÃO	75
6.2. Montando o sistema de arquivos virtual do kernel	75
6.2.1. Criando os dispositivos de nós iniciais	75
6.2.2. Montando e preenchendo o /dev	76
6.2.3. Montando o sistema virtual do kernel	76
6.3. Entrando no ambiente chroot.....	77
6.4. Criando os diretórios	77
6.5. Criando os links simbólicos necessários.....	79
6.6. Linux-2.6.22.5 API Headers	82

6.7. Man-pages-2.63.....	82
6.8. Glibc-2.5.1.....	83
6.8.1. Configurando o Glibc.....	86
6.8.2. Configurando o vinculador dinâmico.....	88
6.9. Reajustando as ferramentas provisórias.....	89
6.10. Binutils-2.17.....	92
6.11. GCC-4.1.2.....	94
6.12. Berkeley DB-4.5.20.....	101
6.13. Sed-4.1.5.....	102
6.14. E2fsprogs-1.40.2.....	103
6.15. Coreutils-6.9.....	105
6.16. Iana-Etc-2.20.....	108
6.17. M4-1.4.10.....	109
6.18. Bison-2.3.....	109
6.19. Ncurses-5.6.....	110
6.20. Procps-3.2.7.....	113
6.21. Libtool-1.5.24.....	114
6.22. Perl-5.8.8.....	114
6.23. Readline-5.2.....	116
6.24. Zlib-1.2.3.....	118
6.25. Autoconf-2.61.....	120
6.26. Automake-1.10.....	120
6.27. Bash-3.2.....	121
6.28. Bzip2-1.0.4.....	123
6.29. Diffutils-2.8.1.....	124
6.30. File-4.21.....	126
6.31. Findutils-4.2.31.....	126
6.32. Flex-2.5.33.....	128
6.33. GRUB-0.97.....	129
6.34. Gawk-3.1.5.....	130
6.35. Gettext-0.16.1.....	132
6.36. Grep-2.5.1a.....	133
6.37. Groff-1.18.1.4.....	134
6.38. Gzip-1.3.12.....	135

6.39. Inetutils-1.5	136
6.40. IPRoute2-2.6.20-070313	138
6.41. Kbd-1.12.....	140
6.42. Less-406	141
6.43. Make-3.81	142
6.44. Man-DB-2.4.4.....	143
6.45. Mktmp-1.5.....	147
6.46. Module-Init-Tools-3.2.2.....	148
6.47. Patch-2.5.4	149
6.48. Psmisc-22.5.....	150
6.49. Shadow-4.0.18.1.....	152
6.49.1. Configurando o Shadow.....	155
6.50. Sysklogd-1.4.1	156
6.50.1. Configurando o Sysklogd.....	157
6.51. Sysvinit-2.86.....	158
6.51.1. Configurando o Sysvinit	159
6.52. Tar-1.18.....	160
6.53. Texinfo-4.8.....	161
6.54. Udev-113	163
6.55. Util-linux-2.12r	165
6.56. Vim-7.1	167
6.56.1. Configurando o Vim	170
6.57. Wget-1.12	171
6.58. Entrando no ambiente chroot – Nova configuração.....	172
CAPÍTULO 7 – Configurando os scripts de inicialização do sistema.....	173
7.1. INTRODUÇÃO	173
7.2. LFS-Bootscripts-6.3.....	173
7.3. Configurando o script setclock.....	174
7.4. Configurando o console Linux	174
7.5. Criando o arquivo /etc/inputrc.....	175
7.6. Os arquivos de inicialização do shell bash.....	177
7.7. Configurando o script localnet	177
7.8. Criando o arquivo /etc/hosts	178
7.9. Configurando o script de rede	179

CAPÍTULO 8 – Tornando o sistema inicializável.....	180
8.1. INTRODUÇÃO	180
8.2. Criando o arquivo /etc/fstab	180
8.3. Linux-2.6.22.5.....	181
8.4. Tornando o sistema LFS inicializável.....	184
CAPÍTULO 9 – O FIM?	186
9.1. Reiniciando o sistema.....	186
CONCLUSÃO	189
REFERÊNCIAS BIBLIOGRÁFICAS	190
Anexo A.....	191
Estudo de Caso	191

LISTA DE FIGURAS

Figura 1: Tela do boot do LiveCD.....	25
Figura 2: Configurar Relógio.....	26
Figura 3: Configurar relógio do sistema.....	26
Figura 4: Configurações regionais.....	27
Figura 5: Confirma ou edita as configurações.....	27
Figura 6: Tela de prompt.	28

Lista de Tabelas

Tabela 1: Comparação de conhecimentos.....	192
--	-----

1. INTRODUÇÃO

Na década de 1960, um dos profissionais dos laboratórios Bell, Ken Thompson, considerado um dos maiores centro de pesquisas do mundo, da AT&T, tenta criar um ambiente de programação mais amigável do que os existentes na época. Utilizando um computador criado pela Digital, criou a primeira versão do Unix. Mais tarde nomes como Dennis Ritchie (criador da linguagem C) prosseguiram com o desenvolvimento das novas versões até 1971 (Unix para redes brasileiras, 1997, p. 1). Em 1973 Dennis Ritchie reescreveu todo o sistema utilizando uma linguagem de alto nível chamada C, criada por ele mesmo. Em 1977 e 1981 a AT&T realizou algumas mudanças particulares lançando o System III e em 1983 mais algumas mudanças realizadas surge então o *System V* que passou a ser vendida e tornando-se o padrão do sistema *Unix*. “Podemos resumir numa frase o ideal do Unix: “fazer realmente o que se propõe a fazer, de uma forma simples e compacta”. E, lembrando o pensamento de Thompson: “Unix é um sistema operacional projetado por programadores para programar..”” (Unix para redes brasileiras, 1997, p. 2). Para ligar o Linux ao Unix uma versão do Unix chamado Minix foi a ponto de partida para tudo. O Minix é um sistema operacional gratuito e com código fonte disponível criado por Andrew S. Tanenbaum para uso educacional. Em 1991 um jovem finlandês chamado Linus Torvalds resolveu criar um sistema melhor que o *Minix*. Para isso resolveu divulgar sua idéia a um grupo de pessoas que utilizavam a *Usenet* (uma espécie de antecessor da internet), Linus escreveu:

Tradução para o português:

Você sente falta dos dias do Minix/1.1 quando homens eram homens e escreviam seus próprios drivers? Você está sem nenhum projeto legal e está ansioso para mexer num sistema operacional que você possa modificar para atender às suas necessidades? Você está achando chato quando tudo funciona no minix? Não ficar mais a noite inteira tentando arrumar um programa legal? Então esta mensagem pode ser para você.

Como eu disse há um mês (?) atrás, eu estou trabalhando numa versão grátis dum similar para o Minix, para computadores AT-386. Ela finalmente atingiu o estágio onde já é usável (apesar de talvez não ser, dependendo do que você quer), e eu estou a fim de colocar (online) o código fonte para uma distribuição melhor. É apenas a versão 0.02 (com mais um patch) mas eu já rodei bash/gcc/gnu-make/gnu-sed/compress dentro dela.

Códigos fontes para este hobby meu podem ser encontradas em nic.funet.fi (128.214.6.100) no diretório /pub/OS/Linux. O diretório também contém alguns arquivos README e um conjunto de arquivos para permitir trabalho no Linux (bash, update e GCC, o que mais você queria? :-). O código-fonte do kernel está disponível por inteiro, porque nenhum do código do Minix foi usado. Os códigos-fontes das bibliotecas são apenas parcialmente abertos, portanto não podem ser distribuídos. O sistema pode compilar "como está" e é provado que funciona. (hehehe) Código-fonte dos programas (bash e gcc) podem ser encontrados no mesmo FTP em /pub/gnu.

PERIGO! AVISO! NOTA! Este código fonte ainda precisa do Minix/386 para compilar (e o gcc-1.4.0, ou o 1.3.7, não teste!) e você precisa do Minix para configurá-lo, então ele ainda não é um sistema por si só para vocês que não tem o Minix. Eu já estou trabalhando nisto. Você também precisa ter um jeito hacker (?) para configurá-lo, então para aqueles torcendo por uma alternativa ao Minix/386, me esqueçam. Ele é atualmente para hackers com interesse no 386 e no Minix.

O sistema precisa de um monitor EGA/VGA e um disco rígido compatível (IDE serve). Se você ainda está interessado, pegue no FTP o readme/relnotes e/ou me mande um e-mail para saber mais.

Eu posso (bem, quase) ouvir vocês perguntando para si mesmos: porquê? O Hurd vai sair em um ano (ou dois, ou em um mês, quem sabe), e eu já tenho o Minix. Este é um programa feito por e para hackers. Eu gostei de fazer ele, e alguém pode começar a olhá-lo e até mesmo modificá-lo às suas necessidades. Ele ainda é pequeno para entender, usar e modificar, e eu estou otimista em relação a algum comentário que vocês tenham a fazer.

Eu também estou interessado em alguém que tenha escrito alguns dos utilitários/bibliotecas para o Minix. Se o seu trabalho pode ser distribuído publicamente (registrado ou mesmo domínio público), eu gostaria de ouvir comentários de vocês, e para que eu possa adicioná-los ao sistema. Eu estou usando o Earl Chews estúdio agora mesmo (obrigado, Earl, por um sistema que funciona), e trabalhos similares seriam bem-vindos. Seus (C)'s obviamente serão mantidos. Me deixe uma mensagem se você quer deixar que a gente use seu código. (Escrito por Emerson Alecrim - Publicado em 30/06/2003 - Atualizado em 12/09/2004).

Seu objetivo não era ganhar dinheiro, mas sim criar um sistema que atendesse suas necessidades por isto adotou a idéia do desenvolvimento em grupo, onde diversos desenvolvedores espalhados pelo mundo participam do projeto. Richard Stallman, fundador da *Free Software Foundation* e criador do projeto GNU (GNU is not Unix) que visa a criação de softwares livres e com qualidade e tinha como principal objetivo desenvolver um sistema operacional baseado no *Unix* mas deveria ser livre, para isso não poderia usar o código fonte da AT&T, o projeto desenvolveu diversas ferramentas mas faltava o item principal: o *kernel*, o projeto de Linus já possuía um *kernel* e este foi utilizado para início ao projeto *Linux*, este nome é a mistura do nome **Linux + Unix**. Com a ajuda de diversos desenvolvedores o sistema cresceu rapidamente, tornando-se um sistema operacional bastante

difundido no mundo inteiro. O *Linux* é somente o *kernel*, porém seu nome é utilizado para definir todos os sistemas que utilizam este *kernel* (Dominando o Linux, 1997, Capítulo 1). Nos dias atuais encontram-se diversas distribuições desenvolvidas com diversos propósitos, como servidores, desktops, entre outros utilizando este *kernel*. Nos dias atuais o Linux é um sistema completo e muito fácil de usar, a ponto de seus usuários não precisarem de um conhecimento muito aprofundado sobre seus sistemas (Dominando o Linux, 1997, Préfacio).

Cada desenvolvedor cria sua distribuição enfatizando suas características mais necessárias de acordo com a aplicação. É possível encontrar centenas de distribuições diferentes, entre elas algumas dezenas mais conhecidas e utilizadas, como por exemplo, *Fedora*, *Slackware*, *Debian*, *OpenSuse* entre outras. É neste ponto que está o problema, a falta de unificação deixa os usuários confusos ao escolher qual o melhor *Linux* que atenda suas necessidades. Algumas distribuições são instaladas com uma quantidade enorme de pacotes, que em muitos casos tornam-se desnecessários para o usuário, outras contêm ferramentas de menos. Este é um dos pontos que este projeto tem o objetivo de atender, ou seja, criar uma distribuição a partir do zero, com as ferramentas mínimas necessárias para o funcionamento do sistema operacional, sendo desta forma um ponto de partida para o próprio usuário controle suas instalações, e configurações, tornando o sistema leve e fornecendo uma base de conhecimento para entender melhor este incrível sistema operacional.

Com o passar dos anos o Linux vem ganhando cada vez mais espaço no mercado mundial como sistema operacional doméstico, mas são tantas distribuições que até mesmo técnicos experientes podem se sentir perdidos na hora de escolher qual melhor distribuição. Várias perguntas podem surgir; qual distribuição oferece melhor desempenho? Qual mais leve? Esta distribuição possui todos os recursos que pretendo usar? Entre outras.

O projeto LFS (Linux From Scratch) no qual este trabalho se baseia, permite que o usuário crie sua própria distribuição Linux, apenas seguindo os procedimentos apresentados no tutorial disponibilizado em seu site. Este trabalho baseia-se no livro Linux From Scratch - Versão 6.1. Desta forma será possível a criação de um sistema Linux funcional apenas seguindo os procedimentos descritos neste trabalho. O

sistema poderá ser personalizado de acordo com a necessidade do usuário, com isso evitando que o sistema consuma recursos com os diversos programas que acompanham as diversas distribuições do mercado que na maioria das vezes não é necessário. Por outro lado o usuário terá um duro trabalho para deixar seu sistema do jeito que ele espera, pois o mesmo estará “enxuto” e todos os programas e recursos que ele deseje devem ser instalados assim como suas dependências.

É claro que a experiência obtida valerá também para que se possa conhecer melhor este sistema operacional, e possa ser aplicadas não apenas neste projeto, mas até mesmo em distribuições prontas, pois todos possuem a mesma base.

2. PRÉ-REQUISITOS

Para o desenvolvimento deste projeto é necessário um conhecimento intermediário sobre Linux e seu funcionamento. Como instalar programas, formatar partições, entre outras. Nas referências bibliográficas constam excelentes fontes de informação para que o leitor possa compreender melhor muitos dos tópicos apresentados neste documento.

3. REQUISITOS DO SISTEMA ANFITRIÃO

Para que este projeto fosse concluído sem maiores problemas alguns requisitos tiveram que ser atendidos:

Kernel 2.6.2 ou maior, necessário para o *Udev* que cria dispositivos dinamicamente;

GCC 3.0 ou maior caso contrário alguns testes realizados falham se o anfitrião não for compilado com a versão 3.0 do compilador *GCC*.

O *Livecdx86 6.3-r2160* utilizado atende a todos esses requisitos. Além disso, já traz consigo todos os pacotes e patches (arquivo que contém correções que são aplicadas ao um arquivo ou programa) necessários para instalação do sistema Linux From Scratch.

4. CONVENÇÕES UTILIZADAS NESTE DOCUMENTO

Para melhor compreensão e acompanhamento deste documento, alguns exemplos tipográficos são apresentados nesta seção, e serão utilizados durante todo o processo de criação deste documento.

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

Este formato indica que deve ser digitado exatamente como esta escrito.

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

Este formato mostra a saída de algum comando na tela. Por exemplo, resultado de algum comando.

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

Neste exemplo apresenta como será o formato de um endereço eletrônico.

5. ESTRUTURA

Este documento esta dividido nas seguintes partes.

5.1. Parte I – INTRODUÇÃO

Nesta parte além de informações gerais sobre o documento, auxilia na compreensão de alguns passos importantes antes de começar a instalação do Linux From Scratch (LFS).

5.2. Parte II – PREPARANDO A CONFIGURAÇÃO

Na parte II é apresenta o processo de construção do sistema LFS, realizando particionamento de disco, preparando o ambiente e pacotes de instalação onde será realizado a compilação das ferramentas provisórias.

5.3. Parte III – CONSTRUINDO O SISTEMA LFS

Na parte III é apresentada a construção do sistema LFS, onde compilará e instalará todos os pacotes necessários, ajustar o ambiente e rotinas de inicialização, resultando em um sistema Linux básico porém funcional, onde servirá de base para continuar com a evolução do sistema, instalando e configurando novos programas.

PARTE I - INTRODUÇÃO

CAPÍTULO 1. INTRODUÇÃO

Para início do projeto deve ser utilizado uma distribuição Linux instalada (Debian, RedHat, SuSE entre outros). Estas distribuições devem atender os requisitos apresentados na seção 3 – REQUISITOS DO SISTEMA ANFITRIÃO.

Uma alternativa pode ser utilizar um *LiveCD* (CD ou DVD contendo um sistema operacional capaz de ser executado sem necessidade de instalação no disco rígido) encontrado no endereço <http://www.linuxfromscratch.org/livecd>.

Uma das vantagens de se utilizar um *LiveCD*, é que o mesmo já possui todos os pacotes e patches (arquivo que contém correções que são aplicadas ao um arquivo ou programa) necessários, sem a necessidade realizar *downloads* destes pacotes. Seguindo esta idéia este documento descreve o projeto LFS tendo como sistema anfitrião o *lfslivecd-x86-6.3-r2160* que pode ser encontrado em:

<ftp://anduin.linuxfromscratch.org/LFS-LiveCD/lfslivecd-x86-6.3-r2160.iso>.

No Capítulo 2 é demonstrado como carregar o sistema anfitrião utilizado na realização do projeto, além disso explica como preparar uma partição com um sistema de arquivos nativo onde será compilado e instalado o sistema LFS. No Capítulo 3 mostra como prepara os pacotes e patches para serem utilizados no projeto. O Capítulo 4 deve ser lido com muita atenção pois nele é realizado a configuração do ambiente de trabalho e processo de montagem do sistema LFS. No Capítulo 5 é realizado a criação de um sistema provisório para que possa ser usado na criação do sistema LFS fina. O Capítulo 6 finaliza a construção do sistema LFS dentro de um ambiente **chroot** (permite transformar o diretório atual em seu diretório raiz), neste momento o sistema ainda não é inicializável, sendo este processo realizado mais adiante. No Capítulo 7 é instalado o LFS-Bootscripts, responsável por criar os *scripts* de inicialização de alguns programas instalados no sistema. No Capítulo 8 o sistema LFS é mostrado os procedimentos para tornar o sistema

inicializável. E finalmente no Capítulo 9 é realizado o procedimento para reiniciar o sistema para que possa ser iniciado e utilizado.

PARTE II – PREPARANDO A CONFIGURAÇÃO

CAPÍTULO 2. CARREGANDO O SISTEMA ANFITRIÃO

O sistema anfitrião necessário para realização do projeto é obtido através de um *LiveCD* como explicado no capítulo anterior. Nesta seção é apresentado como iniciar este sistema.

Basta para isso gravar um *cd-rom* com a imagem baixada do site e configurar o computador para realizar o boot pelo unidade de *cd-rom*.

Ao iniciar o computador com o *cd-rom* na unidade é apresentado uma tela de *boot* do *liveCD* LFS como mostra a figura 1:



Figura 1: Tela do boot do LiveCD.

Nesta tela basta pressionar a tecla *ENTER* e o boot seguirá carregando o sistema operacional. Antes que o sistema esteja totalmente carregado algumas perguntas devem ser respondidas com o decorrer do carregamento, como mostrado nas figuras 2, 3, 4 e 5.

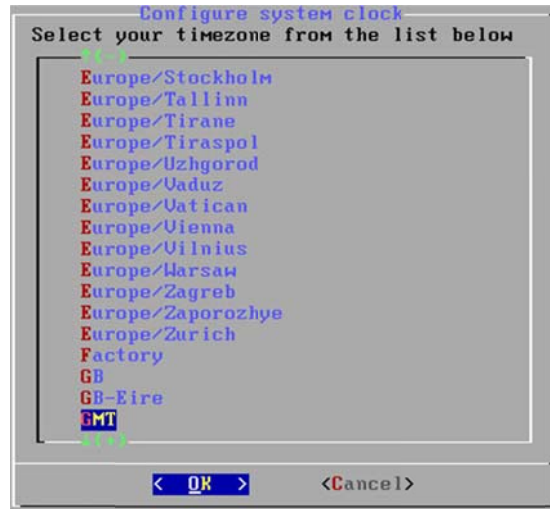


Figura 2: Configurar Relógio.

Na figura 2 o relógio é configurado com a opção *GMT*, em seguida pressione *ENTER* para continuar, em seguida é apresentado uma tela semelhante a figura 3, selecione *Localtime* e pressione *ENTER*.

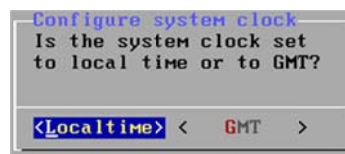


Figura 3: Configurar relógio do sistema.

Continuando como mostrado nas figuras 4 e 5, é necessário selecionar a configuração regional necessário para definir características do sistema, tais como, formato da hora, teclado, data.

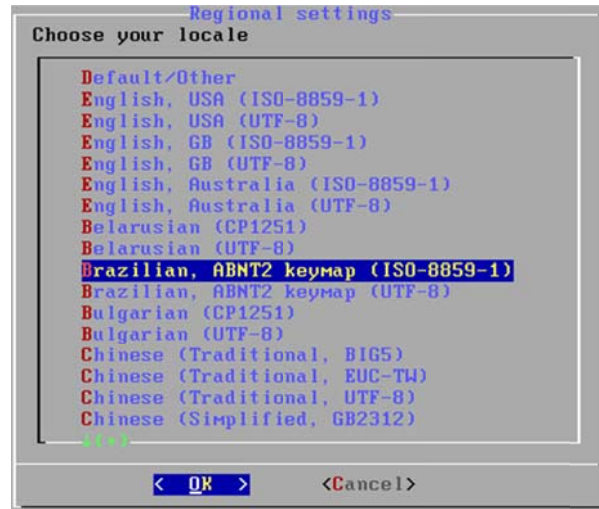


Figura 4: Configurações regionais.

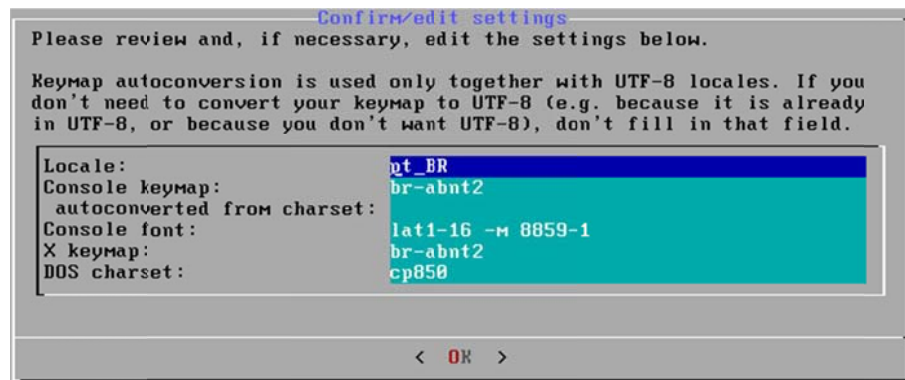


Figura 5: Confirma ou edita as configurações.

Após selecionar e confirmar as opções acima o sistema estará pronto para ser utilizado. O *prompt* é apresentado assim que o usuário pressionar *ENTER* e uma tela como a figura 6 será apresentada.

```

This is the Official Linux From Scratch LiveCD
Created by: Jeremy Huntwork
Maintained by: Alexander Patrakov
Version: x86-6.3-r2160
-----
Please read the /root/README.txt file before using this CD.
All source packages needed to build an LFS system are located in /lfs-sources.
You can find the LFS book in /usr/share/LFS-BOOK-6.3-HTML.
Support questions concerning this CD should be directed to
livecd@linuxfromscratch.org.
Please do NOT send questions regarding building a LFS system or the
instructions in the LFS book to the above address. Instead please
use the lfs-support@linuxfromscratch.org address.
Type greeting at any time to view this message again.
Press Enter to activate this virtual console...
root [ ~ ]#

```

Figura 6: Tela de prompt.

2.2. CRIANDO UMA NOVA PARTIÇÃO

Aqui é criado uma nova partição. O sistema LFS deve ser instalado em uma partição dedicada, para isso é usado o comando *cfdisk*.

```
cfdisk /dev/[xxx]
```

Mude o [xxx] pelo nome do seu dispositivo. Caso não saiba utilizar este programa consulte as páginas *cfdisk(8)* do *man*. Ao criar as partição são dadas designação para elas, como sda1, sda2, anote estas designações pois elas serão necessárias mais tarde.

Duas partições devem ser criadas, neste caso uma denominada sda1 e outra sda2, a partição sda1 contém o sistema LFS, enquanto a partição sda2 é uma partição swap, necessária caso falte memória de acesso aleatório (RAM) suficiente.

2.3. CRIANDO O SISTEMA DE ARQUIVOS

Após a criação das partições estas devem ser formatadas, a partição `sda1` com o sistema de arquivos `ext3` e a partição `sda2` como `swap` com o comando seguinte:

```
mke2fs -j /dev/[xxx]
```

```
mkswap /dev/[yyy]
```

```
swapon /dev/[yyy]
```

Substitua `[xxx]` e `[yyy]` pelo nome de suas partições. Em nosso exemplo `[xxx]` substitui-se por `sda1` e `[yyy]` por `sda2`. O comando `swapon /dev/[yyy]` ativa o uso da partição `swap`.

Agora que as partições foram criadas e formatadas, podem ser acessadas e para isso é criado um diretório com esse propósito. O diretório pode ter o nome que quiser, porém a melhor escolha é um nome que sugira sua intenção, para melhor entendimento. Desta forma é criado um diretório chamado `lfs` e montado a partição neste diretório.

```
mkdir /mnt/lfs
```

```
mount /dev/sda1 /mnt/lfs
```

Uma variável de ambiente com o nome `LFS` contendo o caminho do nosso diretório `lfs` deve ser criado.

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

Para conferir se a variável esta configurada corretamente basta digitar o comando abaixo:

```
echo $LFS
```

A saída dever ser */mnt/lfs*, desta forma quando a variável LFS for usada ela será substituída por seu conteúdo, exemplo:

```
mkdir $LFS/teste
```

O comando acima será substituído no sistema por:

```
mkdir /mnt/lfs/teste
```

Agora tudo esta pronto para instalação dos pacotes, pois temos agora uma partição pronta para a instalação.

CAPÍTULO 3. PACOTES E PATCHES

3.1. INTRODUÇÃO

Neste capítulo é visto como utilizar os pacotes e patches contidos no *LiveCD* baixado da internet. Os pacotes precisam ser colocados em um diretório de trabalho, onde possam ser descompactados para compilação e instalação. Este diretório também deve estar disponível durante todo o processo de instalação do sistema LFS. Este diretório é criado como *writable* e *stick*, isto significa que todos tem permissão de escrito, mas somente o proprietário pode apagar arquivos nele.

```
mkdir $LFS/sources
```

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

Agora podemos transferir os pacotes do *LiveCD* para nosso diretório de trabalho.

```
cp lfs-sources/* $LFS/sources
```

Agora temos todos os pacotes necessários disponíveis no diretório *sources*.

CAPÍTULO 4. PREPARAÇÕES FINAIS

4.1. SOBRE A VARIÁVEL \$LFS

Antes de continuar a variável de ambiente `$LFS` deve ser verificada utilizando o comando abaixo:

```
echo $LFS
```

Caso a saída não seja esta:

```
/mnt/lfs
```

Basta executar o seguinte comando para que a variável seja configurada:

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

4.2. CRIANDO O DIRETÓRIO \$LFS/tools

Um diretório deve ser criado para conter alguns programas compilados e que serão usados na criação de um sistema provisório.

```
mkdir $LFS/tools
```

Um link deste diretório deve ser criado no sistema anfitrião para que as ferramentas como compilador, assembler e linker funcionem corretamente agora e mais adiante quando usaremos o *chroot* na partição LFS.

```
ln -s $LFS/tools /
```

4.3. CRIANDO O USUÁRIO E GRUPO DO SISTEMA LFS

Um usuário chamado *lfs* deve ser criado para que possamos trabalhar no projeto sem correr o risco de cometermos algum erro que comprometa o funcionamento do sistema anfitrião, pois com este usuário as permissões serão restritas.

```
groupadd lfs
```

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

Entenda as opções usadas neste comando:

- s /bin/bash → indica o shell bash como padrão para o usuário lfs.
- g lfs → adiciona o usuário lfs ao grupo lfs.
- m → cria o diretório home do usuário lfs.
- k /dev/null → esta opção com o parâmetro /dev/null impede a cópia dos arquivos-modelo do diretório *skel*.

Uma senha para o usuário *lfs* deve ser criada, e permitir que ele tenha acesso irrestrito aos diretórios *\$LFS/tools* e *\$LFS/sources*.

```
passwd lfs
```

```
chown lfs $LFS/tools
```

```
chown lfs $LFS/sources
```

Uma sessão com o usuário *lfs* pode ser iniciada através do comando abaixo:

```
su - lfs
```

Perceba que agora ao invés do # (sustenido) mostrado no prompt será mostrador o \$ (cifrão), isto indica que o usuário possui limitações.

4.4. CONFIGURANDO O AMBIENTE

Ao iniciar um login *shell* o sistema lê o arquivo de configuração do sistema anfitrião que contém diversos parâmetros de configuração que podem atrapalhar no desenvolvimento do projeto, por isso deve ser criado dois arquivos de configuração contendo apenas parâmetros necessários.

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

```
cat> ~/.bashrc << "EOF"  
  
# desliga a procura por comandos recentemente utilizados  
  
set +h  
  
# define as permissões de criação de arquivos e diretórios.  
  
umask 022  
  
# define a variável de ambiente LFS apontando para /mnt/lfs  
  
LFS=/mnt/lfs  
  
# controla a localização de determinados programas fazendo com que  
  
# suas mensagens sigam as convenções de um determinado país.  
  
LC_ALL=POSIX  
  
# Ajusta o caminho para localização dos comandos  
  
PATH=/tools/bin:/bin:/usr/bin
```

```
# define as variáveis criadas  
  
export LFS LC_ALL PATH  
  
EOF
```

Tendo criado estes dois arquivos agora basta ativar seu perfil.

```
source ~/.bash_profile
```

CAPÍTULO 5. CRIANDO UM SISTEMA PROVISÓRIO

5.1. INTRODUÇÃO

Este capítulo descreve como compilar e instalar as ferramentas provisórias necessárias para instalação do sistema LFS básico do próximo capítulo. Estas ferramentas devem ser instaladas em `$LFS/tools` para que fiquem separadas das ferramentas que devem ser instaladas no próximo capítulo e do sistema anfitrião.

5.2. INSTALANDO PACOTES PROVISÓRIOS

Os programas descritos neste capítulo encontram-se em `$LFS/sources` e são descompactados utilizando o comando `tar`, `bunzip2` ou `gunzip`.

5.2. Binutils-2.17 (Passo 1)

O *Binutils* deve ser o primeiro pacote a ser instalado porque o *GCC* e o *GLibc* executam alguns testes no vinculador dinâmico e no *assembler* para determinar a configuração de suas características.

A documentação recomenda que este pacote seja configurado utilizando um diretório diferente do diretório do código-fonte.

Preparando para a compilação:

```
mkdir binutils-build
```

```
cd binutils-build
```

```
CC="gcc -B/usr/bin/" ../binutils-2.17/configure \
```

```
--prefix=/tools --disable-nls --disable-werror
```

Entenda as opções usadas no comando:

- | | |
|-----------------------------------|---|
| <code>CC="gcc -B/usr/bin/"</code> | → força a utilização do gcc do sistema anfitrião. |
| <code>--prefix=/tools</code> | → instala os programas do <i>Binutils</i> no diretório <i>/tools</i> . |
| <code>--disable-nls</code> | → desabilita a internacionalização que não é necessária neste capítulo. |
| <code>--disable-werror</code> | → evita que a compilação pare por erros de alerta. |

Compilando e instalando o pacote:

```
make
```

```
make install
```

Preparando o vinculador dinâmico que deve ser ajustado mais tarde.

```
make -C ld clean
```

```
make -C ld LIB_PATH=/tools/lib
```

Entenda as opções usadas no comando:

-C ld clean → remove os arquivos compilados do subdiretório ld.

-C ld LIB_PATH=/tools/lib → reconfigura o subdiretório ld, e a variável *LIB_PATH* determina o caminho das bibliotecas.

Atenção

Não remova os diretórios do Binutils ainda, eles serão necessários mais tarde.

5.3. GCC-4.1.2 (Passo 1)

O pacote *GCC* contém os compiladores *C* e *C++*.

A documentação recomenda que este pacote seja configurado utilizando um diretório diferente do diretório do código-fonte.

Preparando para a compilação:

```
mkdir binutils-build
```

```
cd binutils-build
```

```
CC="gcc -B/usr/bin/" ../gcc-4.1.2/configure --prefix=/tools \
```

```
--with-local-prefix=/tools --disable-nls --enable-shared \
```

```
--enable-languages=c
```

Entenda as opções usadas no comando:

`--with-local-prefix=/tools` → pesquisa o *gcc* no diretório */tools*.

`--enable-shared` → Configura as bibliotecas *libgcc_s.so.1* e *libgcc_eh.a*. Isto garante que o pacote *Glibc* produza resultados corretos.

`--enable-languages=c` → somente o compilador C seja configurado.

Compilando e instalando o pacote:

```
make bootstrap
```

```
make install
```

O parâmetro *bootstrap* utilizado no comando *make* faz com que a compilação do GCC seja realizada várias vezes, utiliza a primeira compilação para

realizar a segunda e então uma terceira vez. Compara a segunda compilação com a terceira e verifica se pode ser remontado corretamente.

Agora é criado um vínculo dinâmico do GCC para CC. Alguns programas utilizam o CC ao invés do GCC.

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

5.4. Linux-Libc-Headers-2.6.22.5 API HEADERS

É necessário expor a Application Programming Interface (API) do Kernel Linux para que seja utilizado no sistema.

Instalando os cabeçalhos do kernel:

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

5.5. Glibc-2.5.1

Este pacote contém a biblioteca C principal. Contém diversas rotinas básicas de alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos e assim por diante.

A documentação recomenda que este pacote seja configurado utilizando um diretório diferente do diretório do código-fonte.

Preparando para a compilação:

```
mkdir glibc-build
```

```
cd glibc-build
```

```
../glibc-2.5.1/configure --prefix=/tools \  
--disable-profile --enable-add-ons \  
--enable-kernel=2.6.0 --with-binutils=/tools/bin \  
--without-gd --with-headers=/tools/include \  
--without-selinux
```

Entenda as opções usadas no comando:

- | | |
|------------------------------------|--|
| <code>--disable-profile</code> | → compila sem informações do perfil das bibliotecas. |
| <code>--enable-add-nos</code> | → Usa o add-on NPTL como biblioteca de threading do <i>Glibc</i> . |
| <code>--enable-kernel=2.6.0</code> | → compila o <i>Glibc</i> com suporte ao kernel 2.6.x. |

- `--with-binutils=/tools/bin` → Assegura que não haja erros com os programas do *Binutils* usados durante a compilação do *Glibc*.
- `--without-gd` → evita que o programa *menusagestat* crie vínculos dinâmicos com bibliotecas do sistema anfitrião.
- `--with-headers=/tools/include` → compila o *Glibc* conforme cabeçalhos instalados anteriormente.
- `--without-selinux` → Caso o sistema anfitrião possua a função SELinux, isto é ignorado na compilação, pois nosso sistema não possui esta função.

Compilando o pacote:

```
make
```

Durante a instalação deste pacote mensagens de erro referente a ausência do arquivo `/tools/etc/ld.so.conf` pode aparecer, evite isso com o comando abaixo:

```
mkdir /tools/etc
```

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

Instalando o pacote:

make install

A “internacionalização” deve ser configurada através do comando *locale*, isto garante que algumas características do sistema sejam configuradas corretamente, como datas e até algumas situações mais complexas. Os *locales* apropriados serão instalados mais adiante.

5.6. AJUSTANDO AS FERRAMENTAS PROVISÓRIAS

Com as bibliotecas provisórias C instaladas, a partir de agora todos os pacotes devem ser compiladas utilizando-as. Para isso é preciso ajustar o vinculador dinâmico e os arquivos de especificações.

O vinculador ajustado na primeira compilação do *Binutils* anteriormente deve ser renomeado para que possa ser encontrado e utilizado. Primeiro é necessário realizar um *backup* do arquivos originais e substituí-los com o vinculador ajustado.

```
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
```

Neste ponto é necessário alterar o arquivo de especificações do GCC de modo que aponte para o novo vinculador dinâmico. Para isso é necessário executar o *script* abaixo:

```
gcc -dumpspecs | sed 's@^/lib/ld-linux.so.2@/tools&@g' \  

> `dirname $(gcc -print-libgcc-file-name) `/specs
```

O comando abaixo impede que alguns arquivos do sistema anfitrião consigam entrar no diretório privado de inclusão o GCC.

```
GCC_INCLUDEDIR=`dirname $(gcc -print-libgcc-file-name) `/include &&  

find ${GCC_INCLUDEDIR}/* -maxdepth 0 -xtype d -exec rm -rvf '{}' \; &&  

rm -vf `grep -l "DO NOT EDIT THIS FILE" ${GCC_INCLUDEDIR}/*` &&  

unset GCC_INCLUDEDIR
```

Deve-se agora verificar se todas as ferramentas básicas para compilação e vinculação estão funcionando corretamente, para isso digite o comando abaixo:

```
echo 'main() {}' > dummy.c  

cc dummy.c  

readelf -l a.out | grep ': /tools'
```

Caso tudo esteja funcionando corretamente a saída deve ser esta:

```
[Requesting program interpreter:  

    /tools/lib/ld-linux.so.2]
```

Caso a saída não seja igual a mostrado no quadro acima, algo pode ter dado errado e deve ser corrigido antes de prosseguir. Siga os passos anteriores com cuidado e atenção para tentar localizar o erro.

Caso tudo esteja certo apague os arquivos temporários criados.

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

Com as ferramentas provisórias de compilação e vinculação ajustadas podemos prosseguir com a instalação dos pacotes restantes desta primeira etapa do sistema provisório.

5.7. Tcl-8.4.15

O pacote *Tcl* e os dois pacotes seguintes são responsáveis por dar suporte as ferramentas de testes do *GCC* e *Binutils*.

Preparando para a compilação:

```
cd unix
```

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Instalando o cabeçalho do pacote *Tcl*, o pacote seguinte necessita destes cabeçalhos.

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

Agora deve ser criada uma ligação simbólica:

```
ln -s tclsh8.4 /tools/bin/tclsh
```

5.8. Expect-5.43.0

Contem um programa para dialogar com outros programas interativos, seguindo um script pré-definido.

Diversos *patches* são utilizados na construção do sistema LFS quando necessário.

Este patch é utilizado no pacote *Expect* para correção de uma falha que pode causar falsas falhas durante a execução de testes do pacote *GCC*.

```
patch -Np1 -i ../expect-5.43.0-spawn-1.patch
```

Em seguida forçar o script de configuração a usar */bin/stty* ao invés de */usr/local/bin/stty*. Isto garante que as ferramentas de testes permaneçam sãs para construção final do conjunto de ferramentas.

```
cp configure{,.bak}
```

```
sed 's:/usr/local/bin:/bin:'configure.bak > configure
```

Preparando para compilação:

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

```
--with-tclinclude=/tools/include --with-x=no
```

Entenda as opções usadas no comando:

`--with-tcl=/tools/lib`

→ procura pela instalação provisória do *Tcl* no diretório *tools*.

`--with-tclinclude=/tools/include`

→ procura pelo diretório onde encontra-se os cabeçalhos internos do *Tcl*. Isto evita

que o *configure* cause falhas porque não pode encontrar automaticamente a localização dos cabeçalhos do *Tcl*.

--with-x=no

→ Evita que o script de configuração procure pelo *Tk* (componente GUI do *Tcl*), ou pelas bibliotecas do sistema de janelas *X*, pois ambos ainda não existem no nosso sistema provisório.

Compilando o pacote:

```
make
```

Instalando o pacote:

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

O parâmetro **SCRIPTS=""** previne a instalação de scripts suplementares do *Expect*, que não são necessários.

5.9. DejaGNU-1.4.4

O pacote DejaGNU contém um framework para testar outros programas.

Preparando para compilação:

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

Compilando e instalando o pacote:

```
make install
```

5.10. GCC-4.1.2 (Passo 2)

Com as ferramentas *Tcl*, *Expect* e *DejaGNU* instaladas agora é possível testar o *GCC* e *Binutils* que podem ser recompilados e vinculados a nova *Glibc*. Execute um teste rápido para verificar se os pseudo-terminal interfaces (*PTYs*) do sistema anfitrião estão corretamente configurados.

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

Resposta:

```
The system has no more ptys.
```

```
Ask your system administrator to create more.
```

Caso a mensagem mostrada seja igual a do quadro acima, os *PTYs* do sistema anfitrião não estão configurados corretamente. Neste caso consulte o FAQ do LFS em <http://www.linuxfromscratch.org/lfs/faq.html#no-ptys>.

O script **fixincludes** deve ser suprimido para evitar uma pesquisa no sistema anfitrião dos cabeçalhos do *Glibc* que podem não combinar com os cabeçalhos do *Glibc* do sistema provisório.

```
cp -v gcc/Makefile.in{.,orig}

sed 's@\.\/fixinc\.sh@-c true@' gcc/Makefile.in.orig > gcc/Makefile.in
```

Através do comando *sed* é aplicado o sinalizador *-fomit-frame-pointer* para garantir a consistência da compilação.

```
cp -v gcc/Makefile.in{.,tmp}

sed 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \

> gcc/Makefile.in
```

O patch seguinte é aplicado para corrigir a localização do vinculador dinâmico do GCC.

```
patch -Np1 -i ../gcc-4.1.2-specs-1.patch
```

Um diretório separado deve ser criado para compilação:

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

Antes de configurar o GCC lembre-se de remover todas as variáveis de ambiente que possa alterar as opções de otimização padrão.

Preparando para compilação:

```
../gcc-4.1.2/configure --prefix=/tools \  
--with-local-prefix=/tools --enable-clocale=gnu \  
--enable-shared --enable-threads=posix \  
--enable-__cxa_atexit --enable-languages=c,c++ \  
--disable-libstdcxx-pch
```

Entenda as opções usadas no comando:

- | | |
|-------------------------------------|--|
| <code>--enable-clocale=gnu</code> | → Assegura que o modelo correto de <i>locale</i> seja configurado corretamente para C++. |
| <code>--enable-threads=posix</code> | → Permite manipular as exceções do C++ para os códigos multi-threaded. |

- `--enable-__cxa_atexit` → Permite o uso de `__cxa_atexit` ao invés de `atexit` que registram os destructor de C++ para locais estáticos e objetos globais.
- `--enable-languages=c,c++` → Garante que os compiladores C e C++ estejam configurados.
- `--disable-libstdcxx-pch` → Não compila os cabeçalhos pré-compilados pois ocupam muito espaço e não teremos nenhum uso para eles.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Agora deve ser verificado se todas as ferramentas básicas para compilação e vinculação estão funcionando corretamente, para isso o comando abaixo é utilizado:

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

Caso tudo esteja funcionando corretamente a saída deve ser esta:

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

Caso a saída não seja esta, algo pode estar errado e deve ser corrigido antes de prosseguir. Siga os passos anteriores com cuidado e atenção para tentar localizar o erro.

Caso tudo esteja certo, apague os arquivos temporários criados.

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

5.11. Binutils-2.17 (Passo 2)

Neste passo o *Binutils* deve ser reinstalado. Para isso é necessário criar um diretório separado para compilação do pacote.

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

Preparando para compilação:

```
../binutils-2.17/configure --prefix=/tools \  
--disable-nls --with-lib-path=/tools/lib
```

Entenda as opções usadas no comando:

--with-lib-path=/tools/lib → Indica ao compilado utilize as bibliotecas em */tools/lib*. Isto evita que ele procure no sistema anfitrião.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Agora o vinculado deve ser preparado para a fase de “re-ajuste” mais adiante.

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

5.12. NCurses-5.6

Contém bibliotecas para manipulação de caracteres de tela para criação de menu e painéis.

Preparando para compilação:

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

Entenda as opções usadas no comando:

- without-ada** → Desabilita suporte ao compilador ada que pode estar presente no sistema anfitrião mas não estará disponível no nosso sistema.
- enable-overwrite** → Instala os cabeçalhos do pacote em */tools/include*, garantindo que outros pacotes encontrem-as.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.13. Bash-3.2

Este pacote contém o *Bourne-Again Shell*.

O *patch* aplicado como mostra o quadro abaixo corrige vários erros descobertos desde o lançamento da versão inicial do *Bash-3.1*.

```
patch -Np1 -i ../bash-3.2-fixes-5.patch
```

Preparando para compilação:

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

Entendendo as opções usadas no comando:

--without-bash-malloc → Desativa a alocação de memória do *Bash* que pode causar falha de segmentação e faz uso da

alocação de memória do *Glibc* que é mais estável.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Um link deve ser criado para os programas que usam o *sh* como *shell*.

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

5.14. Bzip2-1.0.4

Este pacote possui programas para comprimir e descomprimir arquivos. O Bzip2 possui melhor compressão de arquivos texto que o Gzip.

O *Bzip2* não possui o script **configure**, basta compilar o pacote:

```
make
```

Instalando o pacote:

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

5.15. Coreutils-6.9

Este pacote possui utilitários para ver e configurar características básicas do sistema.

Preparando para compilar

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

O comando acima não instala o comando *su* porque o usuário *lfs* não tem privilégios para isso, portanto deve ser instalado manualmente com um nome diferente onde poderá ser utilizado para realizar testes no final do sistema com um usuário sem privilégios.

```
cp -v src/su /tools/bin/su-tools
```

5.16. Diffutils-2.8.1

Este pacote contém programas para mostrar diferenças entre arquivos e diretórios.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.17. Findutils-4.2.31

Este pacote contém programas para realizar pesquisas, buscas recursivas em diretórios. Criar e manter uma base de dados para realizar buscas mais rapidamente.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.18. Gawk-3.1.5

Contem programas para manipular arquivos texto.

Preparando para compilação:

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

A correção de um bug no script de configuração do *Gawk* deve ser corrigido, caso contrário ele não consegue detectar alguns aspectos do *locale* do *Glibc*.

```
cat>> config.h << "EOF"  
  
#define HAVE_LANGINFO_CODESET 1  
  
#define HAVE_LC_MESSAGES 1  
  
EOF
```

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.19. Gettext-0.16.1

Este pacote possui utilitários de internacionalização e localização. Eles permitem que os programas sejam compilados com suporte a língua nativa *Native Language Support* (NLS,) habilitando as mensagens de saída na língua nativa do usuário.

Preparando para compilação:

```
cd gettext-tools  
./configure --prefix=/tools --disable-shared
```

Entenda as opções do comando:

`--disable-shared` → Desativa a instalação das bibliotecas compartilhadas, pois não precisamos delas agora.

Compilando o pacote:

```
make -C gnulib-lib
```

```
make -C src msgfmt
```

Instalando o *msgfmt*:

```
cp -v src/msgfmt /tools/bin
```

5.20. Grep-2.5.1a

Este pacote realiza procura em arquivos. Pode ser usado para mostrar a(s) linha(s) que contenham a frase procurada.

Preparando para compilação:

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

```
--disable-perl-regexp
```

Entenda as opções do comando:

`--disable-perl-regexp` → Não realiza a ligação com uma biblioteca Perl *Compatible Regular Expression* (PCRE) que pode estar no sistema anfitrião, mas não em nosso sistema provisório.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.21. Gzip-1.3.12

Este pacote contém programas para comprimir e descomprimir arquivos.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.22. Make-3.81

Este pacote contém o programa para compilar outros pacotes.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.23. Patch-2.5.4

Este pacote modifica ou cria arquivos, aplicando “remendos” especialmente criados pelo programa *diff*.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.24. Perl-5.8.8

O pacote contém o “*Practical Extraction and Report Language*”.

Alguns caminhos são corrigidos para a biblioteca C aplicando o *patch* abaixo:

```
patch -Np1 -i ../perl-5.8.8-libc-2.patch
```

Preparando para compilação:

```
./configure.gnu --prefix=/tools -Dstatic_ext='Data/Dumper Fcntl IO POSIX'
```

Entenda as opções do comando:

`-Dstatic_ext='Data/Dumper Fcntl IO POSIX'` → Constrói um jogo mínimo de extensões estáticas para instalação e testes do pacote *Coreutils* e *Glibc* mais adiante.

Compilando o pacote, somente alguns utilitários precisam ser compilados:

```
make perl utilities
```

Instalando as ferramentas e suas bibliotecas:

```
cp -v perl pod/pod2man /tools/bin
```

```
mkdir -pv /tools/bin/perl5/5.8.8
```

```
cp -Rv lib/* /tools/lib/perl5/5.8.8
```

5.25. Sed-4.1.5

Este pacote contém um editor de fluxo.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.26. Tar-1.18

Contém um programa para empacotamento de arquivos.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

5.27. Texinfo-4.9

Contém programas para ler, escrever e converter páginas info.

Preparando para compilação:

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

Compilando o pacote;

```
make
```

Instalando o pacote:

```
make install
```

5.28. Util-linux-2.12r

Este pacote contém diversos programas responsáveis por montar, desmontar, formatar, particionar, gerenciar discos rígidos, abrir portas *tty* e capturar mensagens do *kernel*.

O script de configuração é alterado para fazer com que o *Util-linux* utilize os cabeçalhos e bibliotecas recentemente instalados em */tools*.

```
sed -i 's@/usr/include@/tools/include@g' configure
```

Preparando para compilação:

```
./configure
```

Compilando algumas rotinas do pacote:

```
make -C lib
```

Somente alguns utilitários serão necessários:

```
make -C mount mount umount
```

```
make -C text-utils more
```

Os seguintes programas devem ser copiados para o diretório de utilitários provisório:

```
cp mount /{,u}mount text-utils/more /tools/bin
```

5.29. Alternado propriedade

A propriedade do diretório `/tools` deve ser alterada, pois este pertence atualmente ao usuário `ifs` que existe apenas no sistema anfitrião. A propriedade deste diretório deve ser do usuário `root`.

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

PARTE III – CONSTRUINDO O SISTEMA LFS

CAPÍTULO 6 – INSTALANDO O SOFTWARE DO SISTEMA BÁSICO

6.1. INTRODUÇÃO

Neste capítulo é usado o ambiente de trabalho configurado anteriormente, para isso usa-se o *chroot* (permite transformar o diretório atual em seu diretório raiz), isto garante que o sistema seja “*enjaulado*”, como se o diretório atual fosse a raiz do sistema e nada fora deste diretório é acessado.

6.2. Montando o sistema de arquivos virtual do kernel

Diversos sistemas de arquivos virtuais são implementados diretamente pelo *kernel* para comunicação do e para o *kernel*. O conteúdo destes sistemas de arquivos reside na memória. É criado então os diretórios onde estes sistemas de arquivos são montados.

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

6.2.1. Criando os dispositivos de nós iniciais

Quando o *kernel* iniciar o sistema ele necessita da presença de alguns dispositivos especiais como, por exemplo, o dispositivo *null* e *console*.

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

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

6.2.2. Montando e preenchendo o /dev

Uma forma de ocupar o diretório */dev* com dispositivos é montar um sistema de arquivos virtual e permitir que os dispositivos sejam criados dinamicamente e sejam detectados ou acessados. Isto é feito geralmente no início da carga do sistema pelo gerenciador dinâmico de dispositivos (*Udev*). Como o sistema ainda não possui o *Udev* deve-se que criar os dispositivos manualmente. Usando uma montagem *bind* que é um tipo de montagem especial que permite que se monte um espelho de outro diretório.

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

6.2.3. Montando o sistema virtual do kernel

Agora montando o sistema virtual do kernel:

```
mount -vt devpts devpts $LFS/dev/pts
```

```
mount -vt tmpfs shm $LFS/dev/shm
```

```
mount -vt proc proc $LFS/proc
```

```
mount -vt sysfs sysfs $LFS/sys
```

6.3. Entrando no ambiente chroot

Entrando agora no ambiente *chroot* para iniciar a configuração final do sistema. Como usuário *root* execute o comando abaixo:

```
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 partir deste ponto o *shell* assume que a variável de ambiente *\$LFS* é o diretório raiz, por isso não há mais necessidade do uso desta variável.

Como ainda não existe o arquivo */etc/passwd* o *bash* exibe a seguinte mensagem no *prompt* “*I have no name!*”.

6.4. Criando os diretórios

A árvore de diretório adotada é baseada na estrutura *Filesystem Hierarchy Standard* (FHS) (disponível em <http://www.pathname.com/fhs>).

```
mkdir -pv /{bin,boot,etc,opt,home,lib,mnt,opt}

mkdir -pv /{media/{floppy,cdrom},sbin,svr,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/{doc,info,locale,man}

mkdir -v /usr/{,local/}share/{misc,terminfo,zoneinfo}

mkdir -pv /usr/{,local/}share/man/man{1..8}

for dir in /usr /usr/local; do

ln -sv share/{man,doc,info} $dir

done

mkdir -v /var/{lock,log,mail,run,spool}

mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

Por padrão os diretórios são criados com a permissão 755, isto garante que todos possam acessar estes diretórios, mas isso não é o ideal, por isso duas mudanças são necessárias, uma no diretório *home* do usuário *root* e outra nos diretórios */tmp* e */var*.

6.5. Criando os links simbólicos necessários

Alguns programas precisam localizar certos programas que ainda não existem em nosso sistema, por esse motivo são criados links simbólicos para resolver este problema, que mais tarde são substituídos pelos programas reais.

```
ln -sv /tools/bin/{bash,cat,echo,grep,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
```

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

O sistema Linux mantém uma lista de sistemas de arquivos armazenada em */etc/mtab*, mas como ainda não é montado nenhum sistema de arquivo no ambiente *chroot*, criaremos um arquivo *mtab* vazio para satisfazer qualquer utilitário que necessite da sua presença.

```
touch /etc/mtab
```

Para que o usuário *root* possa entrar no sistema e ser reconhecido é necessário criar os arquivos */etc/passwd* e */etc/group*.

```
cat> /etc/passwd << "EOF"
```

```
root:x:0:0:root:/root:/bin/bash
```

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

```
EOF
```

```
cat> /etc/group << "EOF"
```

```
root:x:0:
```

```
bin:x:1:
```

```
sys:x:2:
```

```
kmem:x:3:
```

```
tty:x:4:
```

```
tape: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

A *Linux Standard Base* (LSB) disponível em <http://www.linuxbase.org>, recomenda que apenas o grupo *root* com um *Group ID* (GID) com valor igual a 0 (zero) e o grupo *bin* com um *Group ID* (GID) com valor igual a 1 (um) são necessários, os demais grupos e GIDs podem ser escolhidos livremente pelo administrador do sistema.

A criação dos demais grupos se deve a uma convenção utilizada em um grande número de distribuições *Linux* e seguem as exigências de configuração do *Udev*.

Agora um novo *Shell* deve ser iniciado, desta forma o “*I have no name!*” no *prompt* será removido:

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

A opção **+h** utilizada garante que o *bash* não recorde o caminho dos binários executados. Isto assegura o uso dos binários instalados daqui pra frente.

Os programas *login*, *agetty* e *init* (entre outros) gravam logs de informação dos usuários ao entrar no sistema. Para que isso aconteça precisamos criar os arquivos de logs.

```
touch /var/run/utmp /var/log/{bttmp,lastlog,wtmp}
```

```
chgrp utmp /var/run/utmp /var/log/lastlog
```

```
chmod 664 /var/run/utmp /var/log/lastlog
```

6.6. Linux-2.6.22.5 API Headers

O *Linux API Headers* disponibiliza as *APIs* do *kernel* para que seja usado pelo *Glibc*.

O comando *sed* é utilizado para evitar a instalação dos cabeçalhos em */usr/include/scsi*.

```
sed -i '/scsi/d' include/Kbuild
```

Instalando os cabeçalhos:

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

6.7. Man-pages-2.63

Este pacote contém um manual com mais de 3.000 páginas.

Instalando o Man-pages:

```
make install
```

6.8. Glibc-2.5.1

Este pacote contém a biblioteca C principal. Contém diversas rotinas básicas de alocação de memória, busca em diretórios, abertura e fechamento de arquivos, leitura e escrita de arquivos e assim por diante.

Um suporte a Nomes de Domínios Internacionalizados (IDNs) ao *Glibc* deve ser utilizado aplicando um *add-on* (suplemento) para este propósito.

```
tar -xvf ../glibc-libidn-2.5.1.tar.gz
```

```
mv glibc-libidn-2.5.1 libidn
```

No *locale vi_VN.TCVN* o *bash* entra em um loop infinito, para corrigir isto não é instalado este *locale*.

```
sed -i '/vi_VN.TCVN/d' localedata/SUPPORTED
```

As ferramentas provisórias ainda se encontram em */tools*, por esse motivo durante a instalação deste pacote um *script* chamado *test-installation.pl* tenta testar o *Glibc* errado, o problema é corrigido com o comando abaixo:

```
sed -i \  
  
's|libs -o|libs -L/usr/lib -Wl,-dynamic-linker=/lib/ld-linux.so.2 -o|' \  
  
scripts/test-installation.pl
```

O *script ldd* contém uma sintaxe específica do *bash*. Isto deve ser alterado para que não seja instalado outro */bin/sh*.

```
sed -i 's|@BASH@|/bin/bash|' elf/ldd.bash.in
```

A documentação recomenda que este pacote seja configurado utilizando um diretório diferente do diretório do código-fonte.

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

Preparando para compilação:

```
../glibc-2.5.1/configure --prefix=/usr \  
  
--disable-profile --enable-add-ons \  

```

```
--enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

Entenda as opções utilizadas no comando:

`--libexecdir=/usr/lib/glibc` → Modifica a posição do programa **pt_chown** do `/usr/libexec` para `/usr/lib/glibc`.

Compilando o pacote:

```
make
```

Testando os resultados (importante):

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

```
grep Error glibc-check-log
```

Para evitar uma mensagem de erro relacionada a falta do arquivo `/etc/ld.so.conf`, este deve ser criado:

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

Instalando o pacote:

make install

Neste momento os *locales* são instalados, para que se ganhe tempo e espaço é instalado apenas os *locales* necessários. Para isso é usado o comando *localedef*:

```
mkdir -pv /usr/lib/locale  
  
localedef -i de_DE -f ISO-8859-1 de_DE  
  
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro  
  
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 ja_JP -f EUC-JP ja_JP
```

6.8.1. Configurando o Glibc

Um novo arquivo `/etc/nsswitch.conf` é criado para que o *Glibc* trabalhe bem em um ambiente de rede.

```
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
```

Com o comando `tzselect` é determinado o fuso horário local:

```
tzselect
```

Após responder algumas perguntas o *script* gera uma saída com o nome correspondente ao seu fuso horário. Então é criado o arquivo */etc/localtime* com o comando abaixo:

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

Substitua o *<xxx>* pelo nome do fuso horário fornecido pelo *tzselect* (por exemplo, *Canada/Eastern*).

No comando *cp* foi utilizada a opção *--remove-destination* para forçar a remoção da vinculação simbólica já existente.

6.8.2. Configurando o vinculador dinâmico

Por padrão o vinculador dinâmico (*/lib/ld-linux.so.2*) procura em */lib* e */usr/lib* pelas bibliotecas, mas alguns podem se encontrar em outros locais, neste caso é criado o arquivo */etc/ld.so.conf* e nele é colocado os caminhos para outras bibliotecas no sistema.

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

```
# End /etc/ld.so.conf
```

```
EOF
```

6.9. Reajustando as ferramentas provisórias

Com as bibliotecas de C instaladas definitivamente, nosso ambiente é ajustado para que as ferramentas recém-compiladas sejam vinculadas a essas novas bibliotecas.

Um *backup* dos vinculadores em */tools* deve ser realizado e substituído com os vinculadores ajustados anteriormente.

```
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
```

Em seguida é ajustado o arquivo de *specs* do GCC para que ele aponte para o novo vinculador dinâmico.

```
gcc -dumpspecs | sed \
```

```
-e 's@/tools/lib/ld-linux.so.2@/lib/ld-linux.so.2@g' \
```

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

Para verificar se a alteração ocorreu como esperado execute o comando abaixo:

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

Se tudo estiver correto e nenhum erro ocorrer a saída deve ser esta:

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

Deve-se verificar se as configurações estão corretas:

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

Se tudo estiver correto e nenhum erro ocorrer a saída deve ser esta:

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

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

Verifique se o compilador esta localizando corretamente os arquivos cabeçalhos:

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

Este comando deve retornar o seguinte resultado:

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

Verifique agora se o novo vinculador esta usando os caminhos corretos para pesquisa:

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

Se tudo estiver correto e nenhum erro ocorrer a saída deve ser esta:

```
SEARCH_DIR("/tools/i686-pc-linux-gnu/lib")  
  
SEARCH_DIR("/usr/lib")  
  
SEARCH_DIR("/lib");
```

Em seguida verifique se esta usando a *libc* correta:

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

Se tudo estiver correto e nenhum erro ocorrer a saída deve ser esta:

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

Por fim certifique-se que o *Glibc* esta usando o vinculador dinâmico correto:

```
grep found dummy.log
```

Se tudo estiver correto e nenhum erro ocorrer a saída deve ser esta:

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

Tudo verificado e estando tudo correto, apague os arquivos de teste:

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

6.10. Binutils-2.17

Antes da instalação é necessário verificar se os *PTYs* estão funcionando corretamente.

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

Se a mensagem retornada for igual ao texto abaixo, isto significa que os *PTYs* não estão funcionando corretamente:

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

A documentação recomenda que este pacote seja configurado utilizando um diretório diferente do diretório do código-fonte.

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

Preparando para compilação:

```
../binutils-2.17/configure --prefix=/usr \  
  
--enable-shared
```

Compilando o pacote:

```
make tooldir=/usr
```

Neste momento o teste é considerado crítico, portanto não deve ser ignorado:

```
make check
```

Instalando o pacote;

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

Instalando o cabeçalho do arquivo *libiberty* necessários para alguns pacotes:

```
cp -v ../binutils-2.17/include/libiberty.h /usr/include
```

6.11. GCC-4.1.2

O pacote *GCC* contém os compiladores *C* e *C++*.

O comando *sed* é aplicado para suprimir a instalação do *libiberty.a*, pois uma versão do *Binutils* será usada.

```
sed -i 's/install_to_$(INSTALL_DEST) //' libiberty/Makefile.in
```

Para garantir que o compilador seja construído de maneira mais consistente, é aplicado o comando *sed* para garantir que o *flag* `-fomit-frame-pointer` seja utilizado.

```
sed -i 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in
```

O *fixincludes* tenta corrigir erroneamente o sistema de cabeçalhos instalado. Como os cabeçalhos do *GCC-4.1.2* e *Glibc-2.5.1* não requerem esta correção, o *fixincludes* é ignorado utilizando o *script* abaixo:

```
sed -i 's@\.\/fixinc\.sh@c true@' gcc/Makefile.in
```

O *GCC* possui um *script* *gccbug* que verifica a existência do *mktemp*, isto faz com que o *script* utilize menos nomes de arquivos aleatórios. Como o *mktemp* ainda não está instalado sua instalação deve ser simulada.

```
sed -i 's/@have_mktemp_command@/yes/' gcc/gccbug.in
```

A documentação recomenda que este pacote seja configurado utilizando um diretório diferente do diretório do código-fonte.

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

```
cd ../gcc-build
```

Preparando para compilação:

```
../gcc-4.1.2/configure --prefix=/usr \  
--libexecdir=/usr/lib --enable-shared \  
--enable-threads=posix --enable-__cxa_atexit \  
--enable-clocale=gnu --enable-languages=c,c++
```

Compilando o pacote:

```
make
```

Neste momento o teste é considerado crítico, portanto não deve ser ignorado:

Testa os resultados, mas não para nos erros:

```
make -k check
```

Para receber um sumários dos resultados execute:

```
../gcc-4.1.2/contrib/test_summary
```

Algumas falhas inesperadas são difíceis de evitar. A menos que os resultados dos testes sejam muito diferentes dos encontrados no endereço abaixo, é seguro continuar:

http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003

Instalando o pacote:

```
make install
```

Alguns pacotes esperam que o pré-processador C seja instalado no diretório */lib*. Para isso deve ser criado o *link* simbólico:

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

Muitos pacotes utilizam o nome *cc* para chamar o compilador, por isso deve ser criado um *link* simbólico para corrigir isto:

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

Para garantir que a compilação e ligação funcione perfeitamente, deve ser realizado alguns testes de sanidade:

```
echo 'main(){}' > dummy.c
```

```
cc dummy.c -v -Wl,--verbose &> dummy.log
```

```
readelf -l a.out | grep ': /lib'
```

Caso tudo esteja funcionando corretamente e não houver erro algum, a saída deve ser esta:

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

Execute o comando abaixo para garantir que será usado os *startfiles* corretos:

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

Caso tudo esteja funcionando corretamente e não houver erro algum, a saída deve ser esta:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.1.2/../../../../crt1.o succeeded
```

```
/usr/lib/gcc/i686-pc-linux-gnu/4.1.2/../../../../crti.o succeeded
```

```
/usr/lib/gcc/i686-pc-linux-gnu/4.1.2/../../../../crtn.o succeeded
```

O comando abaixo verifica se o compilador esta usando os arquivos de cabeçalho correto:

```
grep -B3 '^ /usr/include' dummy.log
```

Este comando deve resultar com sucesso o seguinte resultado:

```
#include <...> search starts here:  
  
/usr/local/include  
  
/usr/lib/gcc/i686-pc-linux-gnu/4.1.2/include  
  
/usr/include
```

Em seguida deve ser verificado se o novo vinculador está usando os caminhos de pesquisa correta:

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

Caso tudo esteja funcionando corretamente e não houver erro algum, a saída deve ser esta:

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")  
  
SEARCH_DIR("/usr/local/lib")  
  
SEARCH_DIR("/lib")  
  
SEARCH_DIR("/usr/lib");
```

Em seguida deve ser verificado o uso correto da *libc*:

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

Caso tudo esteja funcionando corretamente e não houver erro algum, a saída deve ser esta:

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

Por último é verificado se o GCC o vinculador correto:

```
grep found dummy.log
```

Caso tudo esteja funcionando corretamente e não houver erro algum, a saída deve ser esta:

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

Caso algum dos testes realizados não estejam como demonstrados acima, alguma coisa está errada e antes de continuar deve ser revisto todos os passos anteriores visando corrigir o problema.

Depois que tudo estiver funcionando corretamente, limpar os arquivos de teste:

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

6.12. Berkeley DB-4.5.20

Este pacote contém diversos programas e utilitários usados por outros programas para funções com banco de dados.

Deve ser aplicado um *patch* que corrige alguns erros ao acessar bancos de dados através da sua API Java:

```
patch -Np1 -i ../db-4.5.20-fixes-1.patch
```

Preparando para compilação:

```
cd build_unix  
../dist/configure --prefix=/usr --enable-compat185 --enable-cxx
```

Entenda as opções do comando:

--enable-compat185 → Esta opção constrói a compatibilidade com a API do *Berkeley DB 1.85*.

--enable-cxx → Esta opção habilita a construção das bibliotecas API de C++.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make docdir=/usr/share/doc/db-4.5.20 install
```

O proprietário da documentação instalada deve ser corrigida:

```
chown -Rv root:root /usr/share/doc/db-4.5.20
```

6.13. Sed-4.1.5

Este pacote contém um editor de fluxo.

Preparando para compilação:

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

Entenda as opções do comando:

`--enable-html` → Esta opção constrói a documentação HTML.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.14. E2fsprogs-1.40.2

Este pacote possui ferramentas para manipular o sistema de arquivos *ext2*, também suporta o sistema de arquivo *ext3 journaling*.

Corrige um caminho codificado para */bin/rm* na suíte de testes do *e2fsprogs*.

```
sed -i -e 's@/bin/rm@/tools&@' lib/blkid/test_probe.in
```

A documentação recomenda que este pacote seja configurado utilizando um subdiretório dentro do diretório do código-fonte.

```
mkdir -v build
```

```
cd build
```

Preparando para compilação:

```
../configure --prefix=/usr --with-root-prefix="" \
```

```
--enable-elf-shlibs
```

Entenda as opções do comando:

`--with-root-prefix=""`

→ Esta opção garante que alguns comandos necessários (como por exemplo `e2fsck`) estejam sempre disponíveis no sistema instalando-os em `/lib` e `/sbin`. Se esta opção não for incluída no `configure` estes programas são instalados em `/usr`.

`--enable-elf-shlibs`

→ Cria bibliotecas compartilhadas.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Instalando bibliotecas estáticas e cabeçalhos:

```
make install-libs
```

6.15. Coreutils-6.9

Este pacote possui utilitários para ver e configurar características básicas do sistema.

Um problema conhecido com o programa *uname* deste pacote é que a opção `-p` sempre retorna desconhecido. O seguinte *patch* corrige esse comportamento para arquiteturas Intel:

```
patch -Np1 -i ../coreutils-6.9-uname-1.patch
```

Impedir a instalação dos binários Coreutils que serão instalados por outros pacotes mais tarde:

```
patch -Np1 -i ../coreutils-6.9-suppress_uptime_kill_su-1.patch
```

Corrige alguns erros relacionados com a internacionalização:

```
patch -Np1 -i ../coreutils-6.9-i18n-1.patch
```

Para que os testes adicionados por esta *patch* possam funcionar, as permissões devem ser mudadas:

```
chmod +x tests/sort/sort-mb-tests
```

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Agora a suíte de testes esta pronto para ser executado:

```
make NON_ROOT_USERNAME=nobody check-root
```

O usuário *nobody* é usado para realizar os testes. Alguns testes exigem que este usuário seja membro de mais de um grupo, por esse motivo é criado um grupo temporário e torná-lo membro.

```
echo "dummy:x:1000:nobody" >> /etc/group
```

Executando os testes:

```
su-tools nobody -s /bin/bash -c "make RUN_EXPENSIVE_TESTS=yes check"
```

Removendo o grupo temporário:

```
sed -i '/dummy/d' /etc/group
```

Instalando o pacote:

```
make install
```

Mover os programas para localizações especificadas pelo FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
```

```
mv -v /usr/bin/{false,hostname,ln,ls,mkdir,mknod,mv,pwd,readlink,rm} /bin
```

```
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
```

```
mv -v /usr/bin/chroot /usr/sbin
```

O *script LFS-Bootscripts* depende do *head*, *sleep* e *nice*. Como a partição */usr* pode não estar disponível durante o *boot* estes binários precisam estar na partição raiz:

```
mv -v /usr/bin/{head,sleep,nice} /bin
```

6.16. lana-Etc-2.20

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

O comando a seguir converte os dados brutos para os formatos corretos para */etc/services* e */etc/protocols*.

```
make
```

Instalando o pacote:

```
make install
```

6.17. M4-1.4.10

O pacote *M4* contém um macro processador.

Preparando para instalação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.18. Bison-2.3

Preparando para compilação

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

Caso o *Bison* não esteja no $\$PATH$ (variável de ambiente usado para configurar os caminhos de pesquisa dos programas e bibliotecas) é construído sem suporte a internacionalização das mensagens de erro. Isto é corrigido com a adição seguinte:

```
echo '#define YYENABLE_NLS 1' >> config.h
```

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.19. Ncurses-5.6

Aplice o seguinte patch para corrigir uma série de questões não resolvidas pela ferramenta de análise estática de código, Coverity:

```
patch -Np1 -i ../ncurses-5.6-coverity_fixes-1.patch
```

Preparando para compilação:

```
./configure --prefix=/usr --with-shared --without-debug --enable-widec
```

Entenda as opções do comando:

`--enable-widec` → Essa opção faz com bibliotecas de caracteres-largos (por exemplo, `libncursesw.so.5.6`) seja construído em vez dos normais (por exemplo, `libncurses.so.5.6`). Essas bibliotecas de caracteres-largos são utilizáveis em ambos multibyte tradicional de 8 bits e locais, enquanto as bibliotecas normais só funcionam bem em locais de 8 bits. Bibliotecas de código de caracteres-largos e normais são compatíveis, mas não binariamente compatíveis.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Corrigir a permissão de uma biblioteca que não deve ser executável:

```
chmod -v 644 /usr/lib/libncurses++w.a
```

Mover as bibliotecas para o diretório */lib*:

```
mv -v /usr/lib/libncursesw.so.5* /lib
```

Como as bibliotecas foram movidas alguns links apontam para um arquivo inválido, isto deve ser corrigido:

```
ln -sfv ../lib/libncursesw.so.5 /usr/lib/libncursesw.so
```

Muitas aplicações esperam que o vinculador consiga encontrar as bibliotecas *ncurses* normais e não caracteres-largos. Isto deve ser corrigido:

```
for lib in curses ncurses form panel menu ; do \  
rm -vf /usr/lib/lib${lib}.so ; \  
echo "INPUT(-I${lib}w)" >/usr/lib/lib${lib}.so ; \  
ln -sfv lib${lib}w.a /usr/lib/lib${lib}.a ; \  
done  
ln -sfv libncurses++w.a /usr/lib/libncurses++a
```

Finalmente, certifique-se que aplicações antigas que usam `-lcurses` ainda seja construída em tempo de execução:

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

6.20. Procps-3.2.7

O pacote *procps* contém programas para monitorar processos.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.21. Libtool-1.5.24

O pacote libtool contém uma GNU biblioteca genérica suporte a script. Ela envolve a complexidade do uso de bibliotecas compartilhadas em uma interface consistente e portátil

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.22. Perl-5.8.8

Primeiro crie uma base `/etc/hosts` para ser referenciado em um dos arquivos de configuração do Perl:

```
echo "127.0.0.1 localhost $(hostname)" > /etc/hosts
```

Preparando para compilação:

```
./configure.gnu --prefix=/usr \  
-Dman1dir=/usr/share/man/man1 \  
-Dman3dir=/usr/share/man/man3 \  
-Dpager="/usr/bin/less -isR"
```

Entenda as opções do comando:

`-Dpager="/usr/bin/less -isR"`

→ Corrige um erro na maneira como o *Perl* invoca o programa *less*.

`-Dman1dir=/usr/share/man/man1`

→ Como o programa *Groff* ainda não está instalado o *configure* do *Perl* entende que não necessita do manual *Perl*. Esta opção corrige isto.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.23. Readline-5.2

O pacote *readline* é um conjunto de bibliotecas que oferece linha de comando de edição e recursos de histórico.

Reinstalar *readline* faz com que os arquivos antigos sejam movidos para *<nome-do-arquivo>.old*, isto pode causar erros na ligação do *ldconfig*. Isto é corrigido com os *sed*'s abaixo:

```
sed -i '/MV.*old/d' Makefile.in
```

```
sed -i '{OLDSUFF}/c:' support/shlib-install
```

Readline contém um erro no manuseio de caracteres não-multibyte, corrija isto com o *patch* abaixo:

```
patch -Np1 -i ../readline-5.2-fixes-3.patch
```

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make SHLIB_LIBS=-lncurses
```

Entenda as opções do comando:

SHLIB_LIBS=-lncurses → Esta opção força o *Readline* ligar contra a biblioteca *libncurses* (realmente, *libncursesw*).

Instalando o pacote:

```
make install
```

Mover as bibliotecas estáticas para um local apropriado:

```
mv -v /lib/lib{readline,history}.a /usr/lib
```

Em seguida remova os arquivos `.so` em `/lib` e refazer as ligações em `/usr/lib`:

```
rm -v /lib/lib{readline,history}.so  
  
ln -sfv ../../lib/libreadline.so.5 /usr/lib/libreadline.so  
  
ln -sfv ../../lib/libhistory.so.5 /usr/lib/libhistory.so
```

6.24. Zlib-1.2.3

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

Preparando o pacote para compilação:

```
./configure --prefix=/usr --shared --libdir=/lib
```

Compilando o pacote:

```
make
```

Instalando as bibliotecas compartilhadas:

```
make install
```

O comando anterior instala um arquivo `.so` em `/lib`, deve ser removido e feita a ligação em `/usr/lib`:

```
rm -v /lib/libz.so  
ln -sfv ../../lib/libz.so.1.2.3 /usr/lib/libz.so
```

Construindo as bibliotecas estáticas:

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

Instalando as bibliotecas estáticas:

```
make install
```

Corrigir as permissões das bibliotecas estáticas:

```
chmod -v 644 /usr/lib/libz.a
```

6.25. Autoconf-2.61

O pacote autoconf contém programas para produzir scripts shell que podem configurar automaticamente o código-fonte.

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.26. Automake-1.10

O pacote *Automake* contém programas para gerar arquivos *Makefiles* para uso com *Autoconf*.

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.27. Bash-3.2

O pacote *Bash* contém o *Bourne Again Shell*.

Aplicar correções para vários erros descobertos desde o lançamento do *Bash-3.2*:

```
patch -Np1 -i ../bash-3.2-fixes-5.patch
```

Preparando o pacote para compilação:

```
./configure --prefix=/usr --bindir=/bin \  
--without-bash-malloc --with-installed-readline
```

Entenda as opções do comando:

`--with-installed-readline` → Esta opção diz ao *Bash* para usar as bibliotecas do *readline* que já esta instalada no sistema.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Execute o programa *bash* recém-compilados (substituindo o que está sendo executado):

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

6.28. Bzip2-1.0.4

Este pacote possui programas para comprimir e descomprimir arquivos. O *Bzip2* possui melhor compressão de arquivos texto que o *Gzip*.

Aplicar o *patch* para instalar a documentação:

```
patch -Np1 -i ../bzip2-1.0.4-install_docs-1.patch
```

Preparando o pacote para compilação:

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

```
make clean
```

Entenda as opções do comando:

`-f Makefile-libbz2_so` → Faz com que o *make* utilize um arquivo *Makefile* diferente, neste caso, *Makefile-libbz2_so* que cria a biblioteca dinâmica *libbz2.so* e cria as ligações simbólicas com ela.

Compilando o pacote:

```
make
```

Instalando os programas:

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

Instalando os binários compartilhados no diretório */bin* e fazer as ligações simbólicas necessárias e apagar alguns arquivos:

```
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
```

6.29. Diffutils-2.8.1

Este pacote contém programas para mostrar diferenças entre arquivos e diretórios.

Corrigir com este *patch* um problema com o tratamento de caracteres em branco de acordo com a localidade:

```
patch -Np1 -i ../diffutils-2.8.1-i18n-1.patch
```

O comando acima faz com que a construção de *diffutils* tente reconstruir o *diff.1* fazendo com que a página de manual fique ilegível para o *diffutils*. Isto pode ser corrigido com o comando abaixo:

```
touch man/diff.1
```

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.30. File-4.21

O pacote *File* contém um utilitário para a determinação do tipo de um determinado arquivo ou arquivos.

Preparando o pacote para compilação

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.31. Findutils-4.2.31

Este pacote contém programas para realizar pesquisas, buscas recursivas em diretórios. Criar e manter uma base de dados para realizar buscas mais rapidamente.

Preparando o pacote para compilação:

```
./configure --prefix=/usr --libexecdir=/usr/lib/findutils \  
--localstatedir=/var/lib/locate
```

Entenda as opções do comando:

`--localstatedir=/var/lib/locate` → Esta opção muda a localização do banco de dados de `locate` para `/var/lib/locate` que é compatível com FHS.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Como */usr* podem não estar disponíveis durante os primeiros estágios da inicialização, o programa precisa ser na partição raiz. O script *updatedb* também precisa ser modificado para corrigir um caminho explícito:

```
mv -v /usr/bin/find /bin
```

```
sed -i -e 's/find:=${BINDIR}/find:=\bin/' /usr/bin/updatedb
```

6.32. Flex-2.5.33

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

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

make install

Alguns pacotes precisam encontrar algumas bibliotecas em */usr/lib*. Um link simbólico deve ser criado para resolver isto:

```
ln -sv libfl.a /usr/lib/libl.a
```

Alguns programas não utilizam o *flex* e tentam utilizar o seu antecessor, por isso deve ser criado um *script* chamado *lex* para chamar *flex* em modo de emulação:

```
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
```

6.33. GRUB-0.97

O pacote *GRUB* contém o *Grand Unified Bootloader*.

Comece aplicando o seguinte patch para permitir uma melhor detecção de driver, corrigir alguns problemas do GCC 4.x, e oferecer melhor suporte SATA para alguns controladores de disco:

```
patch -Np1 -i ../grub-0.97-disk_geometry-1.patch
```

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

```
mkdir -v /boot/grub
```

```
cp -v /usr/lib/grub/i386-pc/stage{1,2} /boot/grub
```

6.34. Gawk-3.1.5

Contem programas para manipular arquivos texto.

Em algumas circunstâncias, *Gawk-3.1.5* tenta libertar um pedaço de memória que não foi alocado. Esse bug foi corrigido pela seguinte patch:

```
patch -Np1 -i ../gawk-3.1.5-segfault_fix-1.patch
```

Preparando o pacote para compilação:

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

Devido a um bug no script de configuração, *Gawk* não consegue detectar alguns aspectos de suporte do *locale* na *Glibc*. Este problema é contornado com o comando abaixo:

```
cat >> config.h << "EOF"  
  
#define HAVE_LANGINFO_CODESET 1  
  
#define HAVE_LC_MESSAGES 1  
  
EOF
```

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.35. Gettext-0.16.1

Este pacote possui utilitários de internacionalização e localização. Eles permitem que os programas sejam compilados com suporte a língua nativa *Native Language Support* (NLS,) habilitando as mensagens de saída na língua nativa do usuário.

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.36. Grep-2.5.1a

Este pacote realiza procura em arquivos. Pode ser usado para mostrar a(s) linha(s) que contenham a frase procurada.

O pacote Grep atual tem muitos bugs, principalmente no suporte de *locales*. *RedHat* corrigiu alguns deles com o seguinte patch:

```
patch -Np1 -i ../grep-2.5.1a-redhat_fixes-2.patch
```

Para que os testes adicionados por este *patch* passem, as permissões para o arquivo de teste devem ser corrigidas:

```
chmod +x tests/fmbtest.sh
```

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.37. Groff-1.18.1.4

O pacote *Groff* contém programas para processamento e formatação de texto

Aplice o *patch* que adiciona os dispositivos "*ascii8*" e "*nippon*" para Groff:

```
patch -Np1 -i ../groff-1.18.1.4-debian_fixes-1.patch
```

Muitas fontes de tela não têm aspas simples Unicode e traços nelas. Diga ao Groff para usar os equivalentes ASCII em vez disso:

```
sed -i -e 's/2010/002D/' -e 's/2212/002D/' \  
-e 's/2018/0060/' -e 's/2019/0027/' font/devutf8/R.proto
```

Preparando o pacote para compilação e configurando o tamanho do papel padrão:

```
PAGE=A4 ./configure --prefix=/usr --enable-multibyte
```

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Alguns programas de documentação, tais como xman, não vão funcionar corretamente sem os links simbólicos seguintes:

```
ln -sv eqn /usr/bin/geqn
```

```
ln -sv tbl /usr/bin/gtbl
```

6.38. Gzip-1.3.12

Este pacote contém programas para comprimir e descomprimir arquivos.

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Mova alguns programas que não precisam estar no 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
```

6.39. Inetutils-1.5

O pacote *Inetutils* contém básicos de rede.

Nem todos os programas de *Inetutils* serão instalados, no entanto o pacote insiste em instalar todas as páginas do manual, isto é corrigido com o *patch* seguinte:

```
patch -Np1 -i ../inetutils-1.5-no_server_man_pages-2.patch
```

Preparando o pacote para compilação:

```
./configure --prefix=/usr --libexecdir=/usr/sbin \  
--sysconfdir=/etc --localstatedir=/var \  
--disable-ifconfig --disable-logger --disable-syslogd \  
--disable-whois --disable-servers
```

Entenda as opções do comando:

- `--disable-ifconfig` → Esta opção impede o *Inetutils* de instalar o *ifconfig*, que pode ser usado para configurar interfaces de rede. LFS usa ip do pacote *IPRoute2* para executar essa tarefa.
- `--disable-syslogd` → Esta opção impede a instalação do *System Log Daemon*, que será instalado com o pacote de *Sysklogd*.
- `--disable-whois` → Esta opção incapacita a configuração do cliente *whois*, que está desatualizado.

`--disable-servers` → Esta opção impede a instalação de vários servidores por serem considerados inadequados ao sistema *LFS* e inseguros em alguns casos.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

O programa *ping* deve ser movido para seu lugar correto em conformidade com o sistema *LFS*.

```
mv -v /usr/bin/ping /bin
```

6.40. IPRoute2-2.6.20-070313

Este pacote contém programas básicos e avançados para rede baseada no em *IPv4*.

A instalação de duas paginas do manual está quebrada, isto é corrigido com o *patch* abaixo:

```
sed -i -e '/tc-bfifo.8/d' -e '/tc-pfifo.8/s/pbfifo/bfifo/' Makefile
```

Compilando o pacote:

```
make SBINDIR=/sbin
```

Entenda as opções do comando:

SBINDIR=/sbin → Garante que os binários do *IPRoute2* sejam instalados em */sbin* para que sigam o padrão do sistema *LFS*.

Instalando o pacote:

```
make SBINDIR=/sbin install
```

As bibliotecas de *Berkeley DB* residem em */usr* e usa um banco de dados em */var/lib/arpd/arpd.db*, mas de acordo com o *FHS*, deve estar em */usr/sbin*:

```
mv -v /sbin/arpd /usr/sbin
```

6.41. Kbd-1.12

Este pacote contém um arquivo de tabela de teclas e utilitários de teclado.

O *patch* abaixo corrige um problema com o comportamento do *Backspace* e *Delete*:

```
patch -Np1 -i ../kbd-1.12-backspace-1.patch
```

O *patch* seguinte corrige um erro no *setfont* quando compilado com o GCC-4.1.2:

```
patch -Np1 -i ../kbd-1.12-gcc4_fixes-1.patch
```

Preparando o pacote para compilação:

```
./configure --datadir=/lib/kbd
```

Entenda as opções do comando:

`--datadir=/lib/kbd` → Esta opção garante que os dados do *layout* do teclado esteja na partição raiz ao invés do padrão em `/usr/share/kbd`.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Alguns dos scripts do pacote *LFS-Bootscripts* dependem do *kbd_mode*, *openvt* e *setfont* que podem não estar disponíveis durante os primeiros estágios da inicialização, por isso os binários devem estar na partição raiz.

```
mv -v /usr/bin/{kbd_mode,openvt,setfont} /bin
```

6.42. Less-406

Este pacote contém um visualizador de arquivos de texto.

Preparando o pacote para compilação:

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

Entenda as opções do comando:

`--sysconfdir=/etc` → Esta opção informa aos programas criados pelo pacote *less* para usar os arquivos de configurações em */etc*.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.43. Make-3.81

Este pacote contém o programa para compilar outros pacotes.

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.44. Man-DB-2.4.4

Este pacote contém programas para procurar e visualizar páginas do manual.

Quatro ajustes são realizados para as fontes de *Man-DB*.

O primeiro altera a localização das páginas de manual traduzido que vêm com o *Man-DB*, para que eles sejam acessíveis tanto em tradicionais e *locales UTF-8*:

```
mv man/de{_DE.88591,}
```

```
mv man/es{_ES.88591,}
```

```
mv man/it{_IT.88591,}
```

```
mv man/ja{_JP.eucJP,}
sed -i 's,\*_\*,??,' man/Makefile.in
```

A segunda mudança é a substituição *sed* para apagar as linhas `"/usr/man"` e `"/usr/local/man"` no arquivo `man_db.conf` para evitar resultados redundante quando usando programas como o *whatis*:

```
sed -i -e '\%t/usr/man%d' -e '\%t/usr/local/man%d' src/man_db.conf.in
```

A terceira mudança altera contas de programas que o *Man-DB* deve ser capaz de encontrar em tempo de execução, mas que não foram instalados ainda:

```
cat >> include/manconfig.h.in << "EOF"
#define WEB_BROWSER "exec /usr/bin/lynx"
#define COL "/usr/bin/col"
#define VGRIND "/usr/bin/vgrind"
#define GRAP "/usr/bin/grap"
EOF
```

Finalmente, o patch para corrigir os erros de saída se a página do manual é abortado prematuramente pressionando a tecla 'q'.

```
patch -Np1 -i ../man-db-2.4.4-fixes-1.patch
```

Preparando o pacote para compilação:

```
./configure --prefix=/usr --enable-mb-groff --disable-setuid
```

Entenda as opções do comando:

`--enable-mb-groff` → Diz ao programa *Man* para utilizar os dispositivos de *Groff* “*ascii8*” e “*nippon*” para formatação de páginas de manual não-ISO-8859-1.

`--disable-setuid` → Desabilita o programa *Man* a criar um *setuid* para o usuário *man*.

Compilando o programa:

```
make
```

Instalando o programa:

```
make install
```

Alguns pacotes fornecem páginas Man UTF-8 que esta versão do Man é incapaz de mostrar. O script a seguir permitirá que alguns desses sejam convertidos em codificações esperadas.

```
cat >> convert-mans << "EOF"
```

```
#!/bin/sh -e

FROM="$1"

TO="$2"

shift ; shift

while [ $# -gt 0 ]

do

FILE="$1"

shift

iconv -f "$FROM" -t "$TO" "$FILE" >.tmp.iconv

mv .tmp.iconv "$FILE"

done

EOF

install -m755 convert-mans /usr/bin
```

Caso as páginas de manual seja distribuída como espera o programa *Man-DB*, as páginas podem ser copiadas em */usr/share/man/<language code>*. Ex. Páginas portuguesas devem ser instaladas com o comando abaixo:

```
mkdir -p /usr/share/man/pt

cp -rv man? /usr/share/man/pt
```

Caso as páginas esteja em codificação *UTF-8*, devem ser convertidos com o comando abaixo:

```
mv man7/iso_8859-7.7{,X}
convert-mans UTF-8 ISO-8859-1 man?/*.?
mv man7/iso_8859-7.7{X,}
make install
```

6.45. Mktemp-1.5

O pacote *Mktemp* contém programas usados para criar arquivos temporários seguro em shell scripts

Muitos *scripts* ainda usam o programa obsoleto *tempfile*, que tem funcionalidade semelhante ao *mktemp*. É usado o *patch* abaixo para incluir um invólucro *tempfile*:

```
patch -Np1 -i ../mktemp-1.5-add_tempfile-3.patch
```

Preparando o pacote para compilação:

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

Entenda as opções do comando:

`--with-libc` → Isso faz com que o programa *mktemp* utilize as funções *mkstemp* e *mkdtemp* da biblioteca C do sistema em vez da sua própria.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

```
make install-tempfile
```

6.46. Module-Init-Tools-3.2.2

O pacote *Module-init-tools* contém programas para manipulação de módulos do kernel do Linux maior ou igual à versão 2.5.47.

Primeiro, corrija um problema potencial quando os módulos são especificados usando expressões regulares:

```
patch -Np1 -i ../module-init-tools-3.2.2-modprobe-1.patch
```

Emita os seguintes comandos para realizar os testes (note que o comando *make distclean* é necessário para limpar a árvore de origem):

```
./configure  
make check  
make distclean
```

Preparando o pacote para compilação:

```
./configure --prefix=/ --enable-zlib
```

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.47. Patch-2.5.4

Este pacote modifica ou cria arquivos, aplicando “remendos” especialmente criados pelo programa *diff*.

Preparando para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.48. Psmisc-22.5

O pacote Psmisc contém programas para exibir informações sobre processos em execução.

Preparando o pacote para compilação:

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

Entenda as opções do comando:

`--exec-prefix=""` → Assegura que os binários do *Psmisc* sejam instalados em */bin* em vez de */usr/bin*. Esta é a posição correta de acordo com o *FHS*, porque alguns binários do *Psmisc* são usados pelo pacote *LFS-Bootscripts*.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Não há nenhuma razão para os programas *pstree* e *pstree.x11* ficarem em */bin*. Mova-os para */usr/bin*:

```
mv -v /bin/pstree* /usr/bin
```

Por padrão, o programa *pidof* do *Psmisc* não é instalado. Isto geralmente não é um problema porque ele é instalado mais tarde com o pacote *Sysvinit*, que fornece um programa *pidof* melhor. Se o *Sysvinit* não for usado em um sistema particular, termine a instalação do *Psmisc* criando a seguinte ligação simbólica

```
ln -sv killall /bin/pidof
```

6.49. Shadow-4.0.18.1

O pacote Shadow contém programas para manipular senhas de forma segura.

Corrigir um erro nos programas *useradd* e *usermod* que os impedem de aceitar os nomes de grupo, em vez de números de identificação de grupo para a opção **-g**:

```
patch -Np1 -i ../shadow-4.0.18.1-useradd_fix-2.patch
```

Preparando o pacote para compilação:

```
./configure --libdir=/lib --sysconfdir=/etc --enable-shared \  
--without-selinux
```

Entenda as opções do comando:

`--without-selinux` → Impede que o sistema *SELinux* seja usado, pois ele não está habilitado no sistema *LFS*.

Desativar a instalação do programa de grupos e suas páginas do manual, pois *Coreutils* proporciona uma melhor versão:

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

Desativar a instalação de páginas de manual em chinês e coreano, pois o formato não pode ser lido corretamente pelo Man-DB

```
sed -i -e 's/ ko//' -e 's/ zh_CN zh_TW//' man/Makefile
```

Shadow suporta outras páginas de manual em uma codificação UTF-8. *Man-DB* pode exibir essas codificações recomendada usando o script `convert-mans` instalado:

```
for i in de es fi fr id it pt_BR; do
convert-mans UTF-8 ISO-8859-1 man/${i}/*.*?
done
for i in cs hu pl; do
```

```
convert-mans UTF-8 ISO-8859-2 man/${i}/*.*?
```

```
done
```

```
convert-mans UTF-8 EUC-JP man/ja/*.*?
```

```
convert-mans UTF-8 KOI8-R man/ru/*.*?
```

```
convert-mans UTF-8 ISO-8859-9 man/tr/*.*?
```

Em vez de usar o método *crypt* padrão, use o método mais seguro de criptografia de senha MD5, que também permite senhas com mais de 8 caracteres.

É também necessário alterar o obsoleto `/spool/var/mail` localização das caixas de correio do usuário por `/var/mail` usados atualmente:

```
sed -i -e 's@#MD5_CRYPT_ENAB.no@MD5_CRYPT_ENAB yes@' \  
-e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Um programa deve ser movido ao local adequado:

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

Algumas bibliotecas devem ser movidas para locais mais adequados:

```
mv -v /lib/libshadow.*a /usr/lib
```

```
rm -v /lib/libshadow.so
```

```
ln -sfv ../../lib/libshadow.so.0 /usr/lib/libshadow.so
```

6.49.1. Configurando o Shadow

Para habilitar a proteção de senhas, execute o seguinte comando:

```
pwconv
```

Para habilitar a proteção de senhas de grupos, execute o seguinte comando:

```
grpconv
```

Os arquivos de configuração de Shadow armazenado pelo utilitário `useradd` não é adequado para sistemas LFS. Use os seguintes comandos para alterar o diretório padrão para novos usuários.

```
useradd -D -b /home
```

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

Crie uma senha para o usuário *root* usando o comando abaixo:

```
passwd root
```

6.50. Sysklogd-1.4.1

O pacote *Sysklogd* contém programas que podem criar mensagens de log do sistema.

O seguinte *patch* corrige vários problemas, incluindo um problema quando o *Sysklogd* é construído com o *Kernel* da série 2.6

```
patch -Np1 -i ../sysklogd-1.4.1-fixes-2.patch
```

O seguinte *patch* faz *Sysklogd* tratar bytes no intervalo de 0x80-0x9f literalmente nas mensagens que estão sendo registrados, em vez de substituí-los por códigos octal.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.50.1. Configurando o Sysklogd

Criar um novo arquivo `/etc/syslog.conf` com o comando abaixo:

```
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
```

6.51. Sysvinit-2.86

O pacote *Sysvinit* contém programas para controlar a inicialização, a execução e a finalização do sistema.

Por exemplo, quando o sistema é desligado o *init* envia sinais *TERM* e *KILL* aos processos que ele inicia e os que não devem funcionar de novo. Ao fazer isto ele envia mensagens como “*Sending processes the TERM signal*”, a mensagem será mudada para “*Sending processes started by init the TERM signal*”.

```
sed -i 's@Sending processes@& configured via /etc/inittab@g' \
```

```
src/init.c
```

Compilando o pacote:

```
make -C src
```

Instalando o pacote:

```
make install
```

6.51.1. Configurando o Sysvinit

Criar um novo arquivo `/etc/inittab` com o comando abaixo:

```
cat > /etc/inittab << "EOF"

# Begin /etc/inittab

id:3:initdefault:

si::sysinit:/etc/rc.d/init.d/rc sysinit

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 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
```

6.52. Tar-1.18

Contém um programa para empacotamento de arquivos.

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.53. Texinfo-4.8

Contém programas para ler, escrever e converter páginas info.

O patch abaixo torna válido a volta para as mensagens em Inglês, quando uma localidade multibyte está em uso:

```
patch -Np1 -i ../texinfo-4.9-multibyte-1.patch
```

Texinfo permite que os usuários locais sobrescrevam arquivos arbitrários através de um ataque symlink em arquivos temporários. Aplique o seguinte patch para corrigir isso:

```
patch -Np1 -i ../texinfo-4.9-tempfile_fix-1.patch
```

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Instalando pacotes pertencentes a uma instalação do *TeX*:

```
make TEXMF=/usr/share/texmf install-tex
```

Entenda as opções do comando:

TEXMF=/usr/share/texmf → Esta opção indica o local da raiz da árvore de *TeX* caso seja instalado mais tarde.

Devido a problemas ocasionais nos *Makefiles* de vários pacotes, este pode perder a sincronia com as páginas de informações instaladas no sistema. Para corrigir isto o comando opcional corrigirá isto:

```
cd /usr/share/info
```

```
rm dir
```

```
for f in *
```

```
do install-info $f dir 2>/dev/null

done
```

6.54. Udev-113

O pacote Udev contém programas para criação de dispositivos dinâmicos.

O *tarball* (arquivo *Tar* comprimido) *udev-config* contém arquivos LFS-específico usado para configurar o *Udev*. Descompacte-o no diretório fonte do *Udev*.

```
tar -xvf ../udev-config-6.3.tar.bz2
```

Criação de dispositivos e diretórios que o *Udev* não consegue lidar devido a serem necessárias muito cedo no processo de boot:

```
install -dv /lib/{firmware,udev/devices/{pts,shm}}

mknod -m0666 /lib/udev/devices/null c 1 3

ln -sv /proc/self/fd /lib/udev/devices/fd

ln -sv /proc/self/fd/0 /lib/udev/devices/stdin

ln -sv /proc/self/fd/1 /lib/udev/devices/stdout

ln -sv /proc/self/fd/2 /lib/udev/devices/stderr

ln -sv /proc/kcore /lib/udev/devices/core
```

Compilando o pacote:

```
make EXTRAS=""`echo extras/*/^`
```

Entenda as opções do comando:

EXTRAS=... → Isso cria os binários auxiliares que podem ajudar na escrita de regras personalizadas Udev.

Instalando o pacote:

```
make DESTDIR=/ EXTRAS=""`echo extras/*/^` install
```

Entenda as opções do comando:

DESTDIR=/ → Isso impede que os processos de criação de *Udev* matem todos os processos *udev* que possam ser executados no sistema host.

Por padrão o *Udev* instala alguns poucos arquivos de configuração. Primeiro deve ser instalado os arquivos mais usados de regras previstas pelo *Udev*.

```
cp -v etc/udev/rules.d/[0-9]* /etc/udev/rules.d/
```

Instalando o arquivo de regras específico do *LFS*:

```
cd udev-config-6.3
```

```
make install
```

Instalando a documentação que explica sobre os arquivos de regras específicas do *LFS*:

```
make install-doc
```

Instale a documentação que explicam as regras comumente utilizadas nos arquivos fornecidos pelo *Udev*:

```
make install-extra-doc
```

Instale a documentação que explica como criar regras personalizadas *Udev*:

```
cd ..
```

```
install -m644 -v docs/writing_udev_rules/index.html \
```

```
/usr/share/doc/udev-113/index.html
```

6.55. Util-linux-2.12r

O pacote *Util-linux* contém diversos programas utilitários. Entre eles estão os utilitários para os sistemas de manipulação de arquivos, consoles, partições e mensagens.

O *FHS* recomenda o uso do */var/lib/hwclock* em vez do diretório */etc* como local para o arquivo do *adjtime*. O comando abaixo é executado para torna-lo compatível com o *FHS*.

```
sed -e 's@etc/adjtime@var/lib/hwclock/adjtime@g' \  
  
-i $(grep -rl '/etc/adjtime' .) \  
  
mkdir -pv /var/lib/hwclock
```

Util-linux não consegue compilar as versões mais novas dos cabeçalhos do *kernel Linux*. Os seguintes *patches* corrigem os problemas:

```
patch -Np1 -i ../util-linux-2.12r-cramfs-1.patch \  
  
patch -Np1 -i ../util-linux-2.12r-lseek-1.patch
```

Preparando o pacote para compilação:

```
./configure
```

Compilando o pacote:

```
make HAVE_KILL=yes HAVE_SLN=yes
```

Entenda as opções do comando:

HAVE_KILL=yes → Isso impede que programas sejam finalizados (já instalado por Procps) e instalados novamente.

SLN=yes → Isso impede que o programa *SLN* (a versão estaticamente ligada de *ln* já instalado por *Glibc*) seja instalado novamente.

Instalando o pacote:

```
make HAVE_KILL=yes HAVE_SLN=yes install
```

6.56. Vim-7.1

O pacote *Vim* contém um poderoso editor de textos.

Primeiro, descompacte estes dois arquivos *vim-7.1.tar.bz2* e (opcionalmente) *vim-7.1-lang.tar.gz* no mesmo diretório.

Aplique um *patch* que corrige vários problemas já encontrados e corrigidos pelos mantenedores desde o lançamento desta versão *Vim-7.1*:

```
patch -Np1 -i ../vim-7.1-fixes-1.patch
```

Esta versão do *Vim* instala manuais traduzidos em diretórios que não são pesquisados. Este *patch* faz com que o *Vim* instale os manuais em diretórios que são pesquisados e permite que o *Man-DB* transcodifique a página para o formato desejado em tempo de execução:

```
patch -Np1 -i ../vim-7.1-mandir-1.patch
```

Por fim, alterar o local de instalação padrão do *vimrc* para o diretório */etc*:

```
echo '#define SYS_VIMRC_FILE "/etc/vimrc"' >> src/feature.h
```

Preparando o pacote para compilação:

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

Entenda as opções do comando:

`--enable-multibyte` → Essa opção habilita o suporte para edição de arquivos em codificação de caracteres multibyte.

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

Muitos usuários estão acostumados a usar `vi`, em vez de `vim`. Para permitir a execução do `vim` quando os usuários habitualmente digitarem `vi`, criar uma ligação simbólica, tanto para o binário como para os manuais nas línguas previstas:

```
ln -sv vim /usr/bin/vi  
  
for L in "" fr it pl ru; do  
  
ln -sv vim.1 /usr/share/man/$L/man1/vi.1  
  
done
```

Por padrão, a documentação de *Vim* está instalada em `/usr/share/vim`. A ligação simbólica a seguir permite que a documentação possa ser acessada via `/usr/share/doc/vim-7.1`, tornando-o compatível com a localização da documentação de outros pacotes:

```
ln -sv ../vim/vim71/doc /usr/share/doc/vim-7.1
```

6.56.1. Configurando o Vim

Por padrão o *Vim* é executado em modo incompatível com o *vi*. A opção “*nocompatible*” é incluída abaixo para destacar o fato de um novo modo estar sendo executado.

```
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 o *vim* se comportar de um modo mais útil (o padrão) do que no modo *vi*-compatível. Remova o “no” para manter o velho modo *vi*. O ***set backspace=2*** permite retroceder sobre quebra de linhas, autoindentações e no começo da inserção. A linha ***syntax on*** habilita o destaque de sintaxe do *vim*. Finalmente, o operador ***if*** com a instrução ***set background=dark*** corrige a suposição do *vim guess* 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'
```

6.57. Wget-1.12

O pacote *Wget* contém um utilitário usado para fazer *download* de arquivos da internet.

Este pacote não faz parte dos pacotes encontrados no *Livecd* utilizado para realização deste projeto portanto deve ser baixado da internet no seguinte endereço:

<http://ftp.gnu.org/gnu/wget/wget-1.12.tar.bz2>

Preparando o pacote para compilação:

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

Compilando o pacote:

```
make
```

Instalando o pacote:

```
make install
```

6.58. Entrando no ambiente chroot – Nova configuração

A partir de agora, quando entrar novamente no ambiente *chroot* depois de sair, use o comando *chroot* com as seguintes alterações:

```
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 essas mudanças é que o diretório */tools* não é mais necessário, podendo ser apagado se assim desejar.

Se os sistemas de arquivos virtuais do kernel foram desmontados, manualmente ou através de um reboot, assegurar que os sistemas de arquivos

virtuais do kernel sejam montados quando entrar novamente no chroot. Este processo foi explicado na Seção 6.2.2, "Montando e preenchendo o /dev" e Seção 6.2.3, "Montando o sistema virtual do Kernel".

CAPÍTULO 7 – Configurando os scripts de inicialização do sistema

7.1. INTRODUÇÃO

Neste capítulo é explicado a utilização do *LFS-Bootscripts* para criar os *scripts* de inicialização do sistema, responsáveis pela configuração e inicialização de alguns programas necessários para o funcionamento destes no sistema.

7.2. LFS-Bootscripts-6.3

Instalando o pacote:

```
make install
```

7.3. Configurando o script setclock

O script setclock obtém a data e hora do relógio do computador.

É criado um arquivo em `/etc/sysconfig/clock` com o seguinte comando:

```
cat > /etc/sysconfig/clock << "EOF"

# Begin /etc/sysconfig/clock

UTC=1

# End /etc/sysconfig/clock

EOF
```

Uma boa dica que explica como lidar com o tempo em um sistema *LFS* pode ser encontrado em <http://www.linuxfromscratch.org/hints/downloads/files/time.txt>.

7.4. Configurando o console Linux

Um *script console* é criado para determinar a fonte do terminal e mapa de teclado. Para saber como configurar corretamente a fonte e mapa de teclado de acordo com a língua correta consulte o endereço <http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html>. Com tudo verificado o arquivo é criado com o seguinte comando:

```
cat >/etc/sysconfig/console <<"EOF"  
  
KEYMAP="[arguments for loadkeys]"  
  
FONT="[arguments for setfont]"  
  
EOF
```

7.5. Criando o arquivo `/etc/inputrc`

O arquivo `inputrc` é o arquivo de inicialização do `readline`, com ele é possível modificar o mapa do teclado para situações específicas.

O arquivo abaixo deve ser criado:

```
cat > /etc/inputrc << "EOF"  
  
# Begin /etc/inputrc  
  
# Modified by Chris Lynn <roryo@roryo.dynup.net>  
  
# Allow the command prompt to wrap to the next line  
  
set horizontal-scroll-mode Off  
  
# Enable 8bit input  
  
set meta-flag On  
  
set input-meta On  
  
# Turns off 8th bit stripping
```

```
set convert-meta Off

# Keep the 8th bit for display

set output-meta On

# none, visible or audible

set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions

"\eOd": backward-word

"\eOc": forward-word

# for linux console

"\e[1~": beginning-of-line

"\e[4~": end-of-line

"\e[5~": beginning-of-history

"\e[6~": end-of-history

"\e[3~": delete-char

"\e[2~": quoted-insert

# for xterm

"\eOH": beginning-of-line

"\eOF": end-of-line

# for Konsole

"\e[H": beginning-of-line

"\e[F": end-of-line

# End /etc/inputrc
```

```
EOF
```

7.6. Os arquivos de inicialização do shell bash

O programa */bin/bash* chamado daqui para frente de *shell*, utiliza um conjunto de arquivos para torna-lo mais interativo.

Seus arquivo encontram-se em */etc* para que forneça ajustes globais, caso algum desses arquivos encontrem-se em */home* do usuário seus ajustes sobrepõem os ajustes globais.

É criado o arquivo */etc/profile* com algumas variáveis de ambiente necessárias para dar suporte a língua nativa.

```
cat > /etc/profile << "EOF"

# Begin /etc/profile

export LANG=UTF-8

export INPUTRC=/etc/inputrc

# End /etc/profile

EOF
```

7.7. Configurando o script localnet

Este script é usado para determinar o nome do computador:

```
echo "HOSTNAME=[lfs]" > /etc/sysconfig/network
```

Substitua o [lfs] pelo nome do computador. Você não deve colocar o FQDN (Fully Qualified Domain Name, Nome de Domínio Completamente Qualificado) aqui. Esta informação será colocada no arquivo */etc/hosts*, mais tarde.

7.8. Criando o arquivo */etc/hosts*

Caso uma placa de rede esteja configurada um endereço IP (Internet Protocol) deve ser especificado com um nome *FQDN*, caso ainda não esteja configurado uma placa de rede ainda assim deve ser configurado um nome *FQDN*, para que alguns programas funcionem corretamente.

O arquivo */etc/hosts* é criado com o seguinte comando:

```
cat > /etc/hosts << "EOF"  
  
127.0.0.1 localhost  
  
192.168.1.1 LFS.example.org  
  
EOF
```

7.9. Configurando o script de rede

Um script deve ser criado dentro do diretório `/etc/sysconfig/network-devices`. O comando abaixo cria um arquivo simples no padrão IPv4:

```
cd /etc/sysconfig/network-devices

mkdir -v ifconfig.eth0

cat > ifconfig.eth0/ipv4 << "EOF"

ONBOOT=yes

SERVICE=ipv4-static

IP=192.168.1.1

GATEWAY=192.168.1.2

PREFIX=24

BROADCAST=192.168.1.255

EOF
```

As variáveis acima podem ser alteradas para que contenham o valor correto da rede de trabalho.

Caso o sistema esteja conectado a internet precisa de um arquivo para lidar com o *DNS (Domain Name Service)* que converte nomes de domínio da internet em endereços *IP* e vice e versa.

O arquivo `/etc/resolv.conf` é criado com o comando abaixo:

```
cat > /etc/resolv.conf << "EOF"

# Begin /etc/resolv.conf

domain [Nome de domínio]

nameserver [IP do domínio primário]

nameserver [IP do domínio secundário]

# End /etc/resolv.conf

EOF
```

CAPÍTULO 8 – Tornando o sistema inicializável

8.1. INTRODUÇÃO

Neste capítulo é criado o arquivo `/etc/fstab`, configurado o *kernel* e instalado o *GRUB* para torná-lo inicializável.

8.2. Criando o arquivo `/etc/fstab`

O arquivo *fstab* é utilizado pelo sistema para determinar os pontos de montagem do sistema de arquivos, sua ordem de montagem e verificação de integridade.

```
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 defaults 0 0

sysfs /sys sysfs defaults 0 0

devpts /dev/pts devpts gid=4,mode=620 0 0

shm /dev/shm tmpfs defaults 0 0

# End /etc/fstab

EOF
```

Os campos <xxx> e <yyy> deve ser substituído pela nomenclatura correta da partição criada no sistema, por exemplo, sda1, sda2. O campo <fff> determina o tipo do sistema de arquivo, por exemplo, ext3, ext2, reiserfs.

8.3. Linux-2.6.22.5

A instalação e configuração do *kernel* envolvem alguns procedimentos, configuração, compilação e instalação. Pode-se ler mais sobre o assunto no arquivo *README* que acompanha o código fonte do *kernel* para métodos alternativos de configuração do kernel diferentes deste livro.

Preparando o *kernel* para compilação:

```
make mrproper
```

Isto limpa o *kernel* de qualquer configuração que possa comprometer a configuração.

O *kernel* é configurado através de uma interface baseada em menus, para isto é utilizado o comando abaixo:

```
make LANG=<host_LANG_value> LC_ALL= menuconfig
```

Entenda as opções do comando:

LANG=<host LANG value> LC_ALL= → Define a localidade usada no host, para que o programa consiga desenhar as linhas corretamente no console *Linux*.

O valor <host LANG value> deve ser substituído pelo valor da variável *\$LANG* do sistema anfitrião.

Compilando o pacote:

```
make
```

Instalando os módulos do *kernel*:

```
make modules_install
```

Com a compilação e instalação dos módulos, alguns passos são necessários para completar a configuração do sistema. Alguns arquivos devem ser copiados para o diretório */boot*.

O comando abaixo copia a imagem do *kernel* para o diretório */boot*, levando em conta que a plataforma utilizada é *x86*.

```
cp -v arch/i386/boot/bzImage /boot/lfskernel-2.6.22.5
```

System.map é um arquivo de símbolo para o kernel. Ele mapeia os pontos de função de entrada de cada função na API do kernel, bem como os endereços das estruturas de dados do kernel para o kernel em execução.

```
cp -v System.map /boot/System.map-2.6.22.5
```

O arquivo `.config` contém todas as configurações utilizadas na compilação dos *kernel*, por isto é muito importante guarda-lo para futuras referências.

```
cp -v .config /boot/config-2.6.22.5
```

Instando a documentação do *kernel*:

```
install -d /usr/share/doc/linux-2.6.22.5  
cp -r Documentation/* /usr/share/doc/linux-2.6.22.5
```

8.4. Tornando o sistema LFS inicializável

O sistema agora esta quase pronto, neste ponto é alguns passos devem ser seguidos para tornar o sistema inicializável e com segurança. O *GRUB* (gerenciador de boot) compilado mais cedo é o principal responsável para a execução desta etapa.

O *GRUB* utiliza sua própria nomenclatura para identificação dos discos rígidos no seguinte formato (hdn,m), onde n é o número do disco rígido e o m é o número da partição, e ambos iniciam em zero. Por exemplo *hda1* é ($hd0,0$) e *hdb2* é ($hd1,1$), ao contrário do *Linux* o *GRUB* não considera as unidades de CD-ROM, então caso exista uma unidade de CD-ROM em *hdb* e um disco rígido em *hdc*, para o *GRUB* o disco rígido é identificado como (*hd1*).

Execute o *shell grub*:

```
grub
```

Com atenção as informações acima, determine o nome apropriado para a partição root. No exemplo que segue, supõe-se que a partição root (inicializável) é hda1.

```
root (hd0,0)
```

O comando abaixo diz ao *GRUB* para instalar-se no *MBR* do hda:

```
setup (hd0)
```

Agora pode-se sair do *shell grub* com o comando abaixo:

```
quit
```

Com isto o *GRUB* localizará os arquivos de configuração em */boot/grub*, por isto é criado um arquivo de *menu* contendo as opções de inicialização:

```
cat > /boot/grub/menu.lst << "EOF"  
  
# Begin /boot/grub/menu.lst  
  
default 0
```

```
# Espera 30 segundos antes de iniciar o sistema.  
  
timeout 30  
  
color green/black light-green/black  
  
# Determina a entrada para seleção de inicialização  
  
title LFS 6.3  
  
root (hd0,0)  
  
kernel /boot/lfskernel-2.6.22.5 root=/dev/hda1  
  
EOF
```

O *FHS* espera que o *menu.lst* se encontre em */etc/grub*, o comando abaixo satisfaz esta condição:

```
mkdir -v /etc/grub  
  
ln -sv /boot/grub/menu.lst /etc/grub
```

CAPÍTULO 9 – O FIM?

9.1. Reiniciando o sistema

O sistema agora está pronto para ser reiniciado, mas para isto é necessário executar alguns procedimentos. Primeiro passo é sair do ambiente *chroot*.

```
logout
```

Desmontar o sistema de arquivos virtuais:

```
umount $LFS/dev/pts
```

```
umount $LFS/dev/shm
```

```
umount $LFS/dev
```

```
umount $LFS/proc
```

```
umount $LFS/sys
```

Desmontar o sistema de arquivo do LFS:

```
umount $LFS
```

Finalmente o sistema pode ser reiniciado:

```
shutdown -r now
```

Com o *GRUB* devidamente configurado o *boot* inicializará agora o sistema LFS, e torna-lo pronto para utilização. Deve-se levar em conta que seus recursos são bastante restritos, tendo em vista que o trabalho até aqui cria um sistema funcional, mas também sem muitas das ferramentas encontradas em distribuições prontas encontradas. Isso também significa que o trabalho não termina por aqui, o

projeto *LFS* se estende por outros projetos que dão continuidade a esta primeira etapa do projeto. Por isso é recomendado leituras adicionais encontradas no site <http://www.linuxfromscratch.org>.

CONCLUSÃO

O sistema *Linux* desenvolvido pode ser iniciado perfeitamente, apesar de conter um conjunto mínimo de ferramentas da uma pequena amostra de como este sistema pode ser poderoso. Com a experiência adquirida nos capítulos anteriores é possível expandir o sistema com novas ferramentas e funcionalidades, compilando e instalando os novos pacotes. O *Linux* é um sistema extremamente poderoso, com vasto material sobre o assunto disponível em livros, artigos, internet. Além disso, pode ser configurado de diversas maneiras para atender melhor as necessidades, como por exemplo, um servidor ou um desktop. Muito ainda pode ser realizada a partir deste ponto, uma interface gráfica pode ser instalada e configurada, o *kernel* pode ser configurado de maneira mais eficiente, entre outras possibilidades, podendo chegar a uma distribuição tão eficiente e completa como qualquer outra distribuição com a vantagem do controle de todos os pacotes instalados.

REFERÊNCIAS BIBLIOGRÁFICAS

WELSH, Matt e KAUFMAN, Lar. Dominando o Linux. 2 ed. Rio de Janeiro: Editora Ciência Moderna Ltda., 1997.

ANUNCIACÃO, Heverton. UNIX para Redes Brasileiras. São Paulo: Érica, 1997.

MOTA FILHO, João Eriberto. Linux e seus Servidores. Rio de Janeiro: Editora Ciência Moderna Ltda., 2000.

LFS Project Homepage. Disponível em: <<http://www.linuxfromscratch.org/lfs>>. Acesso em: 04 de abril de 2010.

BLFS Project Homepage. Disponível em: <<http://www.linuxfromscratch.org/blfs>>. Acesso em: 05 de abril de 2010.

Bem-vindo ao Linux From Scratch!. Disponível em: <<http://lfs-br.codigolivre.org.br>>. Acesso em 12 de maio de 2010.

Filesystem Hierarchy Standard. Disponível em: <<http://www.pathname.com/fhs>>. Acesso em 30 de setembro de 2010.

Linux Standard Base (LSB). Disponível em: <<http://www.linuxbase.org>>. Acesso em 14 de setembro de 2010.

BEEKMANS, Gerard. Linux From Scratch rev. 6.1, 2005.

TEAM, BLFS Development, Beyond Linux From Scratch rev. 6.3, 2008.

Linux, Entendendo o Sistema: Um pouco sobre a história do Linux. Disponível em: <<http://www.gdhpress.com.br/entendendo/leia/index.php?p=cap1-2>>. Acesso em 01 de novembro de 2010.

A História do Linux. Disponível em: <<http://www.infowester.com/linux5.php>>. Acesso em 12 de novembro de 2010.

Anexo A

Estudo de Caso

Para avaliação deste trabalho, o mesmo foi utilizado por três indivíduos com características diferentes dos requisitos mínimos exigidos para sua execução. O indivíduo identificado como A1 possui um nível de conhecimento considerado acima de acordo com este trabalho, enquanto o indivíduo identificado como A2 possui conhecimento considerado médio e o indivíduo identificado como A3 possui conhecimento considerado baixo.

A tabela abaixo descreve os passos em que os indivíduos A1 e A2 passaram na execução deste projeto e suas dificuldades:

	A1	A2	A3
Requisitos de conhecimento	Alto	Médio	Baixo
Carregar o sistema anfitrião	Sim	Sim	Sim (necessita ajuda)
Criar Partição e sistema de arquivos	Sim	Sim	Sim (necessita ajuda)
Criar diretórios e arquivos de instalação.	Sim	Sim	Sim (necessita ajuda)
Criar usuários e grupos	Sim	Sim	Sim (necessita ajuda)
Criar o sistema provisório	Sim	Sim (necessita ajuda)	Muita dificuldade

Compilação dos pacotes provisórios	Sim	Sim (necessita ajuda)	Muita dificuldade.
Construir o sistema e configurar o ambiente.	Sim	Sim (necessita ajuda)	Muita dificuldade
Compilar os pacotes	Sim	Sim	Muita dificuldade
Configurar os scripts de inicialização	Sim	Sim (necessita ajuda)	Muita dificuldade
Tornar o sistema inicializável	Sim	Sim (necessita ajuda)	Muita dificuldade

Tabela 1: Comparação de conhecimentos.

De acordo com a tabela acima se pode destacar que o conhecimento mínimo exigido para elaboração deste trabalho é um requisito bastante importante para que a compreensão do mesmo seja mais aproveitada. Mas nada impede que um usuário com um nível médio possa executar este projeto. Infelizmente como desmonstrou a tabela sem um conhecimento apropriado o trabalho torna-se extremamente difícil devendo antes estudar mais sobre o assunto abordado.