



LFS

Suzyanne de Oliveira Queiroz

Retomando

- Capítulo 1 :
Descreve rapidamente como será o processo.
- Capítulo 2 :
Descreve a criação de um sistema de arquivos nativo e partições do Linux.
- Capítulo 3 :
Descreve os pacotes e patches que serão necessários para download e com armazená-los no novo sistema de arquivos.
- Capítulo 4 :
Descreve como configurar um novo ambiente funcional, explicando vários enganos normais a serem evitados.

Capítulo 4

- **Preparando o sistema – Preparações Finais**

Todos os programas que serão compilados na Parte 5 serão instalados no diretório `$LFS/tools`. Isso é para mantê-los separados dos que forem compilados na parte 6, pois eles são apenas temporários e não farão parte do sistema. Isso facilita na hora de removê-los após o sistema estar pronto.

Primeiros passos

- Crie o diretório:
`mkdir -v $LFS/tools`
- Criar um link simbólico para nosso sistema host :
`ln -sv $LFS/tools /`

Criar esse link vai permitir que tudo seja compilado sempre se referindo ao diretório /tools porém sendo feito no diretório /mnt/lfs/tools.



Adicionando o usuário LFS

- Logado como root é fácil danificar o sistema. Então foi criado um usuário LFS para fazer o trabalho. Por padrão, vou usar o indicado pelo livro.

Criando o grupo:

```
groupadd lfs
```

E o usuário:

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

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



- Explicando a linha de comando:
 - s */bin/bash*: Faz com que o “bash” seja o shell padrão do usuário
 - g *lfs*: Adiciona o usuário ao grupo lfs
 - m: Cria o diretório home do usuário
 - k: Este parâmetro previne a cópia de um diretório esqueleto (/etc/skel) mudando a localização de entrada para um dispositivo nulo (/dev/null)
 - lfs*: O nome do usuário

Root e senha

- Para logar com o novo usuário (ao invés de somente trocar do root para ele), temos que criar uma senha:

```
passwd lfs
```

- O lfs dono do diretório \$LFS/tools para garantir o acesso total à ele:

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

- E fazer o mesmo para o \$LFS/sources:

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

Configurando o ambiente do usuário LFS



- Criação de dois arquivos de inicialização para o bash :

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

Isso substitui o shell que está executando por um novo, que contém apenas as variáveis definidas (HOME, TERM e PS1), garantindo que nenhuma variável indesejada (e potencialmente perigosa) do sistema host atrapalhe no ambiente de desenvolvimento.

- A instância de shell é uma instância sem login, então ela não lê o */etc/profile* nem *.bash_profile*, mas lê o *.bashrc*.
- Criamos assim:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL LFS_TGT PATH
EOF
```

- A opção `+h` desabilita a função de hash do bash. Essa função é útil pois ela guarda em uma tabela os comandos executados, reduzindo o tempo de acesso ao PATH e também para encontrar os mesmos executáveis e não novos.
- Configurando a máscara de criação de arquivos para 022 garante que os novos arquivos e diretórios serão escritos (alterados) somente pelo seu criador, porém é de leitura e execução para todos.
- A variável LFS é setada para nosso ponto de montagem.
- A variável LC_ALL controla algumas configurações de alguns programas, relacionadas a localização (país), seguindo certas convenções. Definir essa variável para qualquer coisa diferente de "POSIX" ou "C" pode causar problemas.
- A variável LFS_TGT define uma descrição para a máquina quando estivermos montando nosso *cross compiler* (compilador) e o linker e quanto estivermos compilando nossas ferramentas temporárias.
- Colocando `/tools/bin` na frente do path fará com que todos os programas instalados na Parte 5 sejam encontrados primeiro imediatamente após sua instalação. Esta forma de Path juntamente com a desabilitação do hash limita os riscos de programas antigos serem usados pelo host quando os novos programas estiverem disponíveis.

Capítulo 5

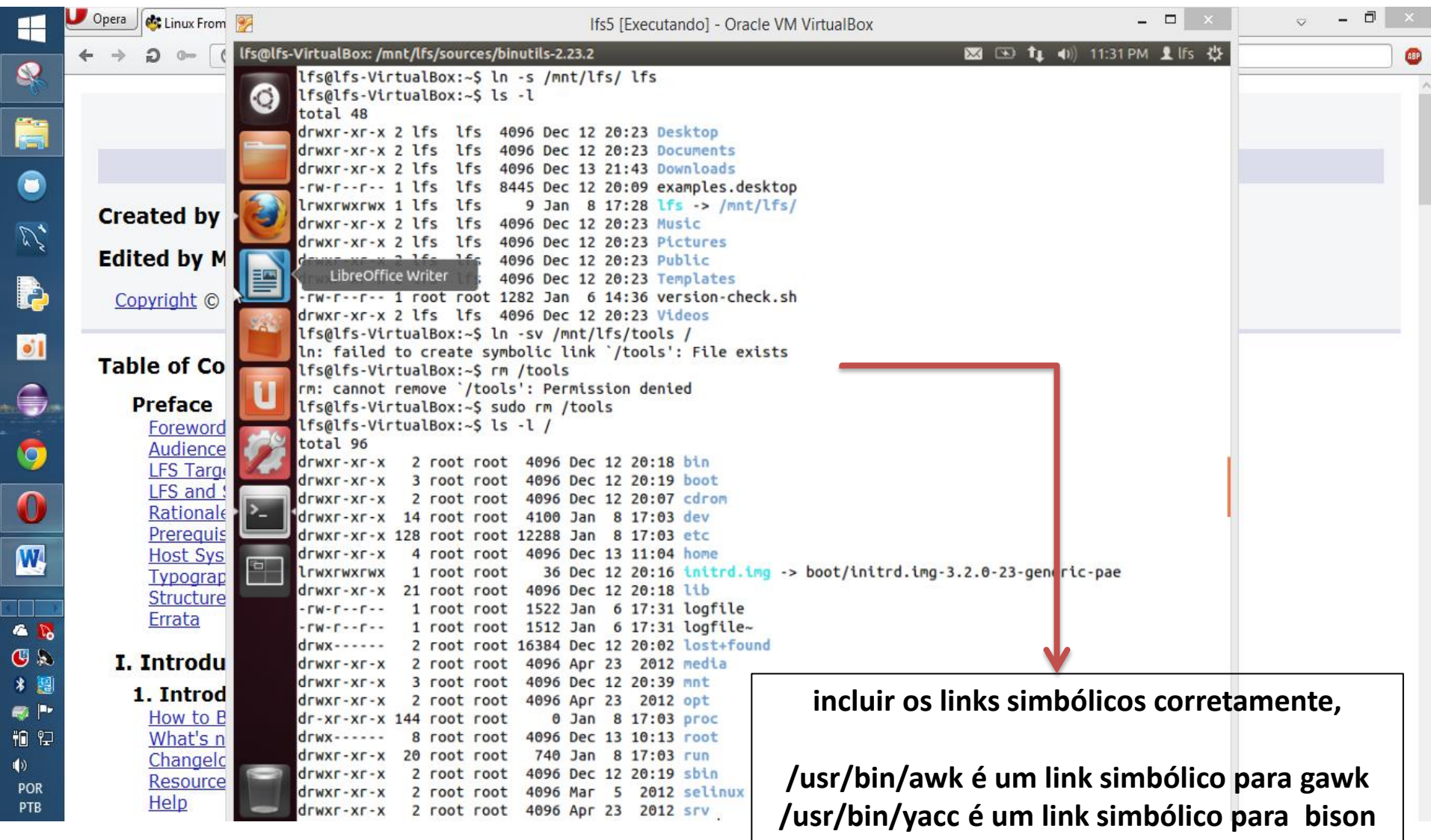
- Primeira coisa antes de compilar os pacotes é estando no usuário LFS fazer o
`export LFS=/mnt/lfs`

Pacotes

- **Linux-3.13.3 API Headers**
- **Glibc-2.19**
- **Libstdc++-4.8.2**
- **Binutils-2.24 - Pass 2**
- **GCC-4.8.2 - Pass 2**
- **Tcl-8.6.1**
- **Expect-5.45**
- **Check-0.9.12**
- **Bash-4.2**
- **Bzip2-1.0.6**
- **Coreutils-8.22**
- **Diffutils-3.3**
- **Findutils-4.4.2**
- **Gawk-4.1.0**
- **Gettext-0.18.3.2**
- **Grep-2.16**
- **Gzip-1.6**
- **M4-1.4.17**
- **Make-4.0**
- **Patch-2.7.1**
- **Perl-5.18.2**

DejaGNU-1.5.1
Ncurses-5.9
Sed-4.2.2
Texinfo-5.2

Primeiro erro



lfs@lfs-VirtualBox: /mnt/lfs/sources/binutils-2.23.2

```
lfs@lfs-VirtualBox:~$ ln -s /mnt/lfs/ lfs
lfs@lfs-VirtualBox:~$ ls -l
total 48
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Desktop
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Documents
drwxr-xr-x 2 lfs lfs 4096 Dec 13 21:43 Downloads
-rw-r--r-- 1 lfs lfs 8445 Dec 12 20:09 examples.desktop
lrwxrwxrwx 1 lfs lfs 9 Jan 8 17:28 lfs -> /mnt/lfs/
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Music
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Pictures
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Public
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Templates
-rw-r--r-- 1 root root 1282 Jan 6 14:36 version-check.sh
drwxr-xr-x 2 lfs lfs 4096 Dec 12 20:23 Videos
lfs@lfs-VirtualBox:~$ ln -sv /mnt/lfs/tools /
ln: failed to create symbolic link '/tools': File exists
lfs@lfs-VirtualBox:~$ rm /tools
rm: cannot remove '/tools': Permission denied
lfs@lfs-VirtualBox:~$ sudo rm /tools
lfs@lfs-VirtualBox:~$ ls -l /
total 96
drwxr-xr-x 2 root root 4096 Dec 12 20:18 bin
drwxr-xr-x 3 root root 4096 Dec 12 20:19 boot
drwxr-xr-x 2 root root 4096 Dec 12 20:07 cdrom
drwxr-xr-x 14 root root 4100 Jan 8 17:03 dev
drwxr-xr-x 128 root root 12288 Jan 8 17:03 etc
drwxr-xr-x 4 root root 4096 Dec 13 11:04 home
lrwxrwxrwx 1 root root 36 Dec 12 20:16 initrd.img -> boot/initrd.img-3.2.0-23-generic-pae
drwxr-xr-x 21 root root 4096 Dec 12 20:18 lib
-rw-r--r-- 1 root root 1522 Jan 6 17:31 logfile
-rw-r--r-- 1 root root 1512 Jan 6 17:31 logfile~
drwx----- 2 root root 16384 Dec 12 20:02 lost+found
drwxr-xr-x 2 root root 4096 Apr 23 2012 media
drwxr-xr-x 3 root root 4096 Dec 12 20:39 mnt
drwxr-xr-x 2 root root 4096 Apr 23 2012 opt
dr-xr-xr-x 144 root root 0 Jan 8 17:03 proc
drwx----- 8 root root 4096 Dec 13 10:13 root
drwxr-xr-x 20 root root 740 Jan 8 17:03 run
drwxr-xr-x 2 root root 4096 Dec 12 20:19/sbin
drwxr-xr-x 2 root root 4096 Mar 5 2012 selinux
drwxr-xr-x 2 root root 4096 Apr 23 2012 srv
```

Created by
Edited by M
Copyright ©

Table of Co
Preface
[Foreword](#)
[Audience](#)
[LFS Target](#)
[LFS and](#)
[Rationale](#)
[Prerequisites](#)
[Host System](#)
[Typography](#)
[Structure](#)
[Errata](#)

I. Introdu
1. Introdu
[How to Build](#)
[What's New](#)
[Changelog](#)
[Resources](#)
[Help](#)

incluir os links simbólicos corretamente,
/usr/bin/awk é um link simbólico para gawk
/usr/bin/yacc é um link simbólico para bison

Pacotes



Created by

Edited by M

Copyright ©

Table of Co

Preface

[Foreword](#)

[Audience](#)

[LFS Target](#)

[LFS and S](#)

[Rationale](#)

[Prerequis](#)

[Host Sys](#)

[Typograp](#)

[Structure](#)

[Errata](#)

I. Introdu

1. Introd

[How to B](#)

[What's n](#)

[Changelo](#)

[Resource](#)

[Help](#)

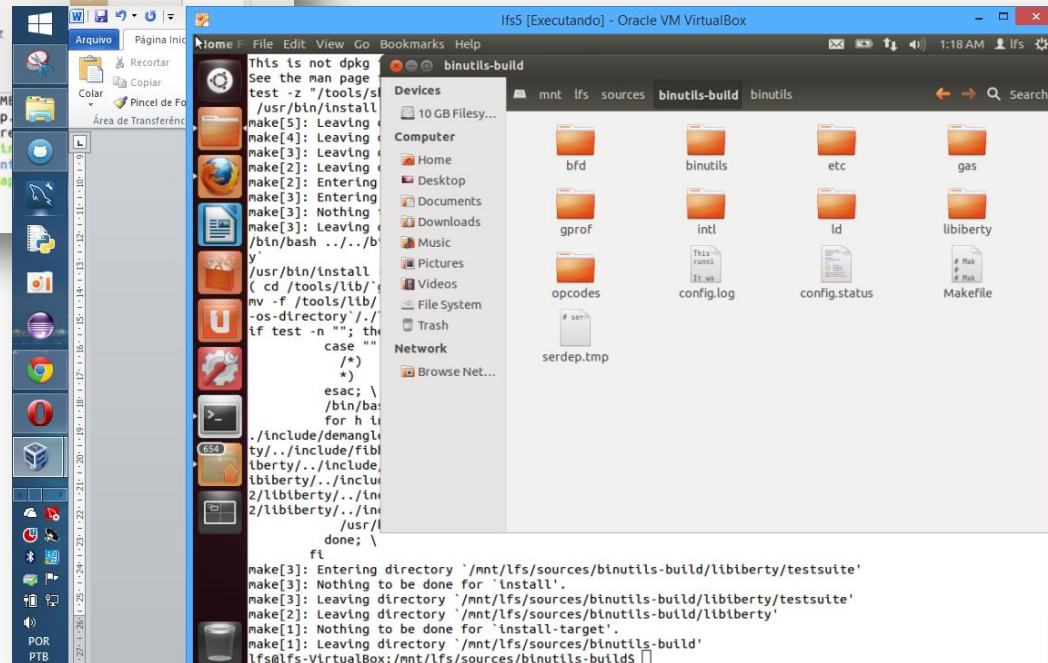
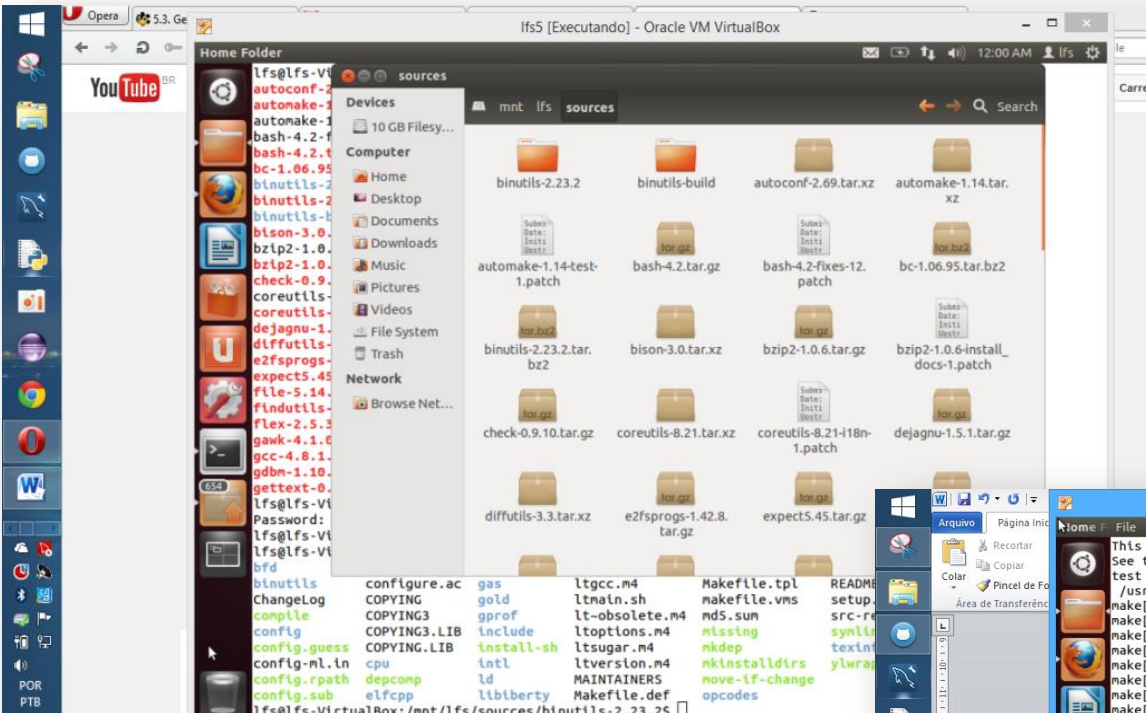
lfs@lfs-VirtualBox: /mnt/lfs/sources/binutils-2.23.2

```
lfs@lfs-VirtualBox:/mnt$ cd lfs
lfs@lfs-VirtualBox:/mnt/lfs$ ls
logfile  sources  tools
lfs@lfs-VirtualBox:/mnt/lfs$ cd sources
lfs@lfs-VirtualBox:/mnt/lfs/sources$ ls
autoconf-2.69.tar.xz      glibc-2.18.tar.xz      patch-2.7.1.tar.xz
automake-1.14.tar.xz      gmp-5.1.2.tar.xz       perl-5.18.1-libc-1.patch
automake-1.14-test-1.patch  grep-2.14.tar.xz       perl-5.18.1.tar.bz2
bash-4.2-fixes-12.patch   groff-1.22.2.tar.gz    pkg-config-0.28.tar.gz
bash-4.2.tar.gz          grub-2.00.tar.xz       procs-ng-3.3.8.tar.xz
bc-1.06.95.tar.bz2       gzip-1.6.tar.xz        psmisc-22.20.tar.gz
binutils-2.23.2          iana-etc-2.30.tar.bz2  readline-6.2-fixes-1.patch
binutils-2.23.2.tar.bz2  inetutils-1.9.1.tar.gz  readline-6.2.tar.gz
binutils-build          iproute2-3.10.0.tar.xz  sed-4.2.2.tar.bz2
bison-3.0.tar.xz         kbd-1.15.5-backspace-1.patch  shadow-4.1.5.1.tar.bz2
bzip2-1.0.6-install_docs-1.patch  kbd-1.15.5.tar.gz      syslogd-1.5.tar.gz
bzip2-1.0.6.tar.gz       kmom-14.tar.xz         systemd-206.tar.xz
check-0.9.10.tar.gz      less-458.tar.gz        sysvinit-2.88dsf.tar.bz2
coreutils-8.21-i18n-1.patch  lfs-bootscripts-20130821.tar.bz2  tar-1.26-manpage-1.patch
coreutils-8.21.tar.xz    libpipeline-1.2.4.tar.gz  tar-1.26.tar.bz2
e2fsprogs-1.42.8.tar.gz   libtool-2.4.2.tar.gz   tcl8.6.0-src.tar.gz
expect5.45.tar.gz        linux-3.10.10.tar.xz   texinfo-5.1.tar.xz
file-5.14.tar.gz         m4-1.4.16.tar.bz2     texinfo-5.1-test-1.patch
findutils-4.4.2.tar.gz   make-3.82.tar.bz2     tzdata2013d.tar.gz
flex-2.5.37.tar.bz2      make-3.82-upstream_fixes-3.patch  udev-lfs-206-1.tar.bz2
gawk-4.1.0.tar.xz        man-db-2.6.5.tar.xz   util-linux-2.23.2.tar.xz
gcc-4.8.1.tar.bz2        man-pages-3.53.tar.xz  vim-7.4.tar.bz2
gdbm-1.10.tar.gz         md5sums               wget-list
gettext-0.18.3.tar.gz    mpc-1.0.1.tar.gz      xz-5.0.5.tar.xz
lfs@lfs-VirtualBox:/mnt/lfs/sources$ cd binutils-
binutils-2.23.2/ binutils-build/
lfs@lfs-VirtualBox:/mnt/lfs/sources$ cd binutils-
binutils-2.23.2/ binutils-build/
lfs@lfs-VirtualBox:/mnt/lfs/sources$ cd binutils-2.23.2/
lfs@lfs-VirtualBox:/mnt/lfs/sources/binutils-2.23.2$ ls
bfd          configure     etc          libtool.m4   Makefile.in   README
binutils     configure.ac  gas          ltoc.m4      Makefile.tpl  README-maintainer-mode
```

Binutils



- Extrair o pacote
- Criar uma pasta build
- Preparar para compilação
- Make install
- E observar se deu tudo certo na pasta TOOLS.



O que é o Make, makefile e Make install?



- Make é, de forma geral, um automatizador de tarefas: ele possibilita que você crie scripts para tarefas comuns e os nomeie. Assim, quando precisar executar dada tarefa, pode executar apenas *make <nome_da_tarefa>*. Essas tarefas ficam no arquivo Makefile. O comando *make install* executa a tarefa chamada install.

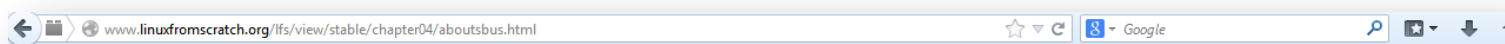
O que é o configure?

- O comando *configure* faz o trabalho inicial: configura paths, detecta o shell utilizado, verifica as dependências etc. Esse comando é um script gerado automaticamente e, após ser executado, gera o Makefile com as configurações específicas do seu sistema

SBU

- O LFS pode ser construído em vários sistemas diferentes, é impossível fazer uma previsão.
- Por exemplo, a compilação da biblioteca Glibc leva em torno de 20 minutos em sistemas rápidos, mas pode levar até 3 dias em sistemas lentos. Então o livre usa o SBU (Standard Build Unit).
- O SBU funciona da seguinte forma: O primeiro pacote a ser compilado e instalado é o Binutils. O tempo que demorar para compilar e instalar esse pacote será usado como referência para o SBU. Os próximos tempos serão referenciados relativos a este tempo.
- Por exemplo. Se o Binutils levou 10 minutos para compilar e instalar e o SBU de outro pacote for de 5, então este pacote leva em torno de 50 minutos para ficar pronto.

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



Linux From Scratch - Version 7.5 Chapter 4. Final Preparations

[Prev](#)
Setting Up the Environment

[Up](#)
[Home](#)

[Next](#)
About the Test Suites

4.5. About SBUs

Many people would like to know beforehand approximately how long it takes to compile and install each package. Because Linux From Scratch can be built on many different systems, it is impossible to provide accurate time estimates. The biggest package (Glibc) will take approximately 20 minutes on the fastest systems, but could take up to three days on slower systems! Instead of providing actual times, the Standard Build Unit (SBU) measure will be used instead.

The SBU measure works as follows. The first package to be compiled from this book is Binutils in [Chapter 5](#). The time it takes to compile this package is what will be referred to as the Standard Build Unit or SBU. All other compile times will be expressed relative to this time.

For example, consider a package whose compilation time is 4.5 SBUs. This means that if a system took 10 minutes to compile and install the first pass of Binutils, it will take *approximately* 45 minutes to build this example package. Fortunately, most build times are shorter than the one for Binutils.

In general, SBUs are not entirely accurate because they are used to give an estimate of how long it might take to install a package.

To view actual timings for a number of specific machines, see <http://www.linuxfromscratch.org/~sbu/>.

Note

For many modern systems with multiple processors, the SBU measure is not accurate.

