

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RN
DIRETORIA ACADÊMICA DE GESTÃO E INFORMÁTICA
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

*Ao professor Leonardo Minora
para a disciplina de Sistemas Operacionais*

**RELATÓRIO DE DESENVOLVIMENTO DO SISTEMA
OPERACIONAL LINUX, BASEADO NO LIVRO LINUX
FROM SCRATCH (LFS).**

Suzyanne de O. Queiroz

07/02/2014

INTRODUÇÃO

Para composição parcial da nota da disciplina de Sistemas Operacionais foi proposta a compilação de um sistema operacional Linux. Foi usado como ferramenta de ajuda e estudo o tutorial do Linux From Scratch, disponível em : <http://www.linuxfromscratch.org/lfs/view/stable/>. O livro fornece instruções necessárias para a criação do sistema Linux, enfocando as vantagens de utilizar tal sistema, tais como a disposição dos diretórios, a instalação dos scripts e questões de segurança. Em suma, o tutorial pretende dar ao usuário a liberdade do arranjo do sistema final, compilando, do zero, todas as partes do projeto a partir de códigos-fonte. Antes de iniciado o tutorial, a leitura do Guia Foca Linux fez-se necessária, a fim esclarecer comandos e parâmetros básicos para o entendimento do processo de criação do sistema, para usuários que não possuem domínio prévio do shell. Foi recomendada pelo professor, a utilização de uma máquina virtualizada como sistema anfitrião. Tal medida traz vantagens no salvamento e recuperação de etapas desenvolvidas e não afetar os recursos de hardware da máquina hospedeira.

DESENVOLVIMENTO

Para hospedar o novo sistema operacional, foi virtualizado, em primeira instância, o Ubuntu 12.04 32 bits. O KERNEL do sistema anfitrião foi compilado com gcc e é compatível com as exigências. A verificação foi feita, em shell, através do comando:

cat /proc/version

Para hospedar o LFS, foi necessária preparação de uma partição dedicada com tamanho maior que o sistema final já compilado, devido à necessidade de espaço adicional durante a compilação dos pacotes.

Na primeira tentativa de particionamento, através do comando fdisk, a tabela de partições do sistema anfitrião foi alterada, incluindo a referência para partição de boot, acarretando em quebra do sistema após sua reinicialização. Para resolver o problema, foi adicionado um disco virtual VDI de 40 Giga e este, particionado em dois: uma para a instalação do sistema e outra para o swap e, em seguida, suas respectivas formatações.

Para ver se o sistema host possuía todas as versões apropriadas, e a capacidade de compilar programas, um script de shell foi criado e rodado. As versões do bison, gawk, g++, patch e makeinfo (4.13) estavam pendentes e foram devidamente instaladas.

O ponto de montagem do sistema foi definido em **/mnt/lfs**. A variável de sistema \$LFS foi configurada para apontar a este diretório e utilizada durante todo o processo de montagem:

Export LFS=/mnt/lfs

Em seguida, com acesso *root*, o diretório de trabalho *sources* foi definido e a pasta modificada a partir do comando *chmod*, a fim de torná-la "writable" e Sticky. Um diretório Sticky, mesmo provendo permissão de escrita a vários usuários, somente pode ter arquivos deletados pelo usuário que o criou:

mkdir -v \$LFS/sources
chmod -v a+wt \$LFS/sources

Packages e patches foram baixados utilizando o comando *wget*. Este deve baixar arquivos de uma rede de forma não iterativa, utilizando protocolos http, https, ftp e proxies http. O comando é geralmente associado à função de download recursivo.

wget -i http://www.linuxfromscratch.org/lfs/view/stable/wget-list -P \$LFS/sources

Na verificação dos packages baixados, observou-se a ausência do package Shadow (4.1.5.1), posteriormente localizado na internet e adicionado ao diretório *sources*. Também utilizado o recurso do comando *wget*, foram baixados os patches para correção de possíveis erros.

Todos os programas a serem compilados no sistema temporário devem ser instalados em *\$LFS/tools*. Foi então criado o diretório, a partir do comando *mkdir*.

Também em modo *root*, foi criado um link simbólico direcionando o termo */tools* para *\$LFS/tools*, a fim de encaminhar corretamente ao local de compilação:

In -sv \$LFS/tools /tools

Para evitar deslizes ou danos irrecuperáveis através das permissões *root*, foi recomendado montar os pacotes com um usuário sem tais privilégios. Os comandos a seguir foram executados para, respectivamente, adicionar um grupo de *users*, adicionar um usuário chamado *lfs2*, vinculando *bash* como *shell* padrão e configuração de senha:

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

Para que o novo usuário tivesse acesso às pastas necessárias para a compilação dos pacotes, o comando *chown* foi aplicado aos diretórios *\$LFS/tools* e *\$LFS/sources*. Nesta etapa, tivemos dificuldade de executar o comando, por problemas com o link simbólico executado incorretamente. Segue a saída do comando:

```
failed to change ownership of `/mnt/lfs/sources' to lfs  
chown: cannot access `/mnt/lfs/tools': No such file or directory
```

Os links simbólicos foram refeitos e o comando reexecutado com sucesso:

```
changed ownership of `/mnt/lfs/sources' from root to lfs2  
changed ownership of `/mnt/lfs/tools from root to lfs2
```

Antes da construção do sistema temporário, a configuração do ambiente foi feita, como usuário *lfs2*, impedindo que alguma variável não desejada do sistema hospedeiro tenha acesso indevido ao ambiente de compilação. A seguir, o arquivo *.bashrc* foi criado para ser lido no lugar do *.bash_profile*. O novo arquivo desativa a função **hash** do **bash**, obrigando o **shell** a sempre procurar o **PATH** quando um programa for rodar. Também é criada uma máscara de usuário que dá permissões de escrita apenas ao dono, mas permite leitura e execução por qualquer um. A variável **LC_ALL** padroniza o ambiente nas definições do POSIX e evita problemas de saída e retorno ao ambiente. Por fim, colocar a pasta */tools/bin* no início do **PATH**, combinada com a desativação da **hash**, faz com que os programas a serem compilados sejam imediatamente acessados pelo **shell** após a instalação.

Durante a etapa de compilação do sistema temporário, encontramos algumas dificuldades em relação à configuração dos links simbólicos que deveriam estar seguramente definidos. Como não encontramos, no livro, definição anterior de tais links, executamos o comando **In** para as referências citadas, mas isso gerou redundância e loops infinitos, prejudicando o andamento da etapa. Apenas o pacote *binutils-2.23.2* foi compilado, com auxílio do professor. A compilação do pacote seguinte, o **gcc**, não pôde ser concluída por dificuldade na interpretação das saídas de erro, tanto no **shell** quanto nos arquivos **config.log**.

ANEXO 1

Anexo 1 – Cópia do arquivo 'logfile' contendo saídas de comandos importantes

**# Comentários serão precedidos com '#'
O restante do conteúdo foi gerado por redirecionamento de saída**

13:12:2013

In: failed to create symbolic link `/tools': File exists

**# Havia uma pasta já criada com o nome de referencia para o
link a ser criado. Pasta removida e comando refeito:**

`/tools' -> `/mnt/lfs/tools'

**chown: cannot access `/mnt/lfs/tools': No such file or directory
failed to change ownership of `/mnt/lfs/tools' to lfs
chown: cannot access `/mnt/lfs/sources': No such file or directory
failed to change ownership of `/mnt/lfs/sources' to lfs
chown: cannot access `/mnt/lfs/tools': No such file or directory
failed to change ownership of `/mnt/lfs/tools' to lfs
In: failed to create symbolic link `/tools': File exists
`/tools' -> `/tools'
chown: cannot dereference `/tools': Too many levels of symbolic links
failed to change ownership of `/tools' from 4294967295 to lfs2
chown: cannot access `/sources': No such file or directory
failed to change ownership of `/sources' to lfs2
chown: cannot access `/mnt/lfs/tools': No such file or directory
failed to change ownership of `/mnt/lfs/tools' to lfs2
ownership of `/mnt/lfs/tools' retained as lfs2
changed ownership of `/mnt/lfs/sources' from root to lfs2**

06/01/2014

**mkdir -p -- /tools /tools
mkdir: cannot create directory `/tools': File exists
mkdir: cannot create directory `/tools': File exists
make[1]: *** [installdirs] Error 1
make[1]: Leaving directory `/mnt/lfs/sources/binutils-build'
make: *** [install] Error 206/01/2014
mkdir -p -- /tools /tools
mkdir: cannot create directory `/tools': File exists
mkdir: cannot create directory `/tools': File exists
make[1]: *** [installdirs] Error 1
make[1]: Leaving directory `/mnt/lfs/sources/binutils-build'
make: *** [install] Error 2**

Anexo 2 – Cópia do arquivo de rascunho para acompanhamento das etapas do projeto

Relatorio de acompanhamento da instalação do sistema operacional Linux no Daruma

12/11

Início da leitura do livro LINUX FROM SCRATCH (VERSÃO EM PORTUGUES)

Fonte de apoio: <http://goo.gl/7pHA8y>

Tentativa de emular o sistema operacional montado LFS realizada com sucesso.

13/11

Instalação do SO UBUNTU na máquina virtual para prosseguir com os passos do LFS Book.

O KERNEL do sistema anfitrião foi compilado com gcc e é compatível com as exigências.

25/11

Criação de arquivo de texto utilizando o "cat"

```
cat < teste.txt << "EOF"
```

Trecho relevante:

"O programa chroot (change root) é usado para entrar em um ambiente virtual e inicializar um novo shell cujo o diretório de raiz será definido na partição do LFS. Isto é muito similar a reinicializar e a instruir o kernel para montar a partição LFS como a partição root. O sistema não reinicializa realmente, mas faz um chroot porque criar um sistema inicializável requer o trabalho adicional que não é necessário neste momento. A vantagem principal de "chrooting" é permitir o uso do sistema anfitrião enquanto o LFS estiver sendo configurado. Enquanto espera a compilação de algum pacote terminar, o usuário pode abrir um console virtual diferente (VC) ou o desktop X e continuar usando seu computador normalmente."

Criando uma nova partição:

Trecho de estudo sobre particionamento de disco

Consulta de partições do sistema através do comando

```
sudo fdisk -l
```

Em seguida, configuração de novas partições através de

```
sudo cfdisk
```

Duas novas particões foram criadas, dividindo a memória em partes iguais, sendo uma delas bootavel.

Observação: Após alteração das partições do da Virtul Machine (VM), a linha de comando segue apresentando erros de inicialização e não reconhece o vínculo do sudoers para autenticação do comando sudo.

Maquina virtual quebrada. Realização do processo de instalação de novo SO virtualizado (nome: tads2).

26/11

Continuação de estudos sobre particionamento.

Criação de novo disco virtualizado pela Virtualbox com tamanho de 40G

divisão do novo disco em 4 partições através do
fdisk /dev/sdb

27/11

2.3 Formatação de partições e arquivo de sistema

Partição /dev/sdb1 formatada em ext4,
criação e formatação de swap de 2GB em dev/sdb5
mkswap /dev/sdb5

Obs.: sdb2 deletada

2.4 Montando nova partição

escolhido o ponto de montagem /mnt/lfs para exportar a variável de ambiente LFS
export LFS=/mnt/lfs

criado o ponto de montagem do arquivo de sistema
mkdir -pv \$lfs
e montado em
mount -v -t ext4 /dev/sdb1 \$LFS

02/12/2013

Criação de diretório de trabalho a partir do comando:

mkdir -v \$LFS/sources

Em seguida, pasta modificada a partir do comando chmod, a fim de torná-la "writable" e Sticky. Um diretório Sticky, mesmo provendo permissão de escrita a vários usuários, somente pode ter arquivos deletados pelo usuário que o criou.

chmod -v a+wt \$LFS/sources

Packages e patches baixados utilizando o comando wget. Este fornece baixa arquivos de uma rede de forma não iterativa, utilizando protocolos http, https, ftp e proxies http. O comando é geralmente associado à função de download recursivo.

Devido a problemas com a máquina onde a virtualização estava sendo rodada, optou-se pela troca de sistema operacional, no entanto a máquina virtual criada não foi recuperada

12/12/2013

Nova máquina virtual criada (Ubuntu 12.04) e passos efetuados anteriormente refeitos.

seguindo para a o capítulo 4: Final Preparations.

16/12/2013

criação do diretório tools em \$LFS

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

através do comando ln, um link simbólico no sistema principal para a pasta recém-criada tools será feito

Criação de novo grupo de usuários no qual será criado um usuário sem privilégios root.

```
groupadd lfs2
```

```
useradd -s /bin/bash -g lfs2 -m -k /dev/null lfs2
```

Criação de novo usuário 'lfs2' (no grupo lfs2) que será usado para a compilação dos pacotes, por não ter permissões root, evitando danificar o pc.

o usuário lfs2 tem acesso irrestrito garantido às pastas tools e sources através do comando:

```
chown -v lfs2 $LFS/tools
```

```
chown -v lfs2 $LFS/sources
```

4.4 Configurando o ambiente

definição de dois arquivos base para o shell do interpretador de comandos 'bash':

```
cat > ~/.bash_profile << "EOF"
```

```
exec env -i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash
```

```
EOF
```

```
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
```

Para finalizar a preparação, o comando a seguir é executado para informar ao shell em andamento as diretrizes passadas no arquivo de parâmetro:

```
source ~/.bash_profile
```

20/12/2013

Na etapa 5.3, detalhes importantes foram enfatizados como a criação de links simbólicos para bash, gawk e bison.

A definição dos links simbólicos causou um loop que impede a procedimentos básicos como a função 'man'.

A máquina será descartada e Snapshot do dia 16/12 (etapa 4.4) será reutilizado.

2 BIMESTRE

10/02/2014 a 18/03/2014

Foi preciso voltar para o capítulo 4 para entender melhor como seria a construção do sistema provisório LFS, a proposta desse capítulo é permitir um ambiente de trabalho mais conveniente para o usuário de como um ambiente mínimo do LFS faria.

O próximo passo é compilar os pacotes para construir esse sistema provisório e esses arquivos compilados serão instalados no LFS/tools \$ diretório para mantê-los do diretório do sistema hospedeiro que será construído mais na frente.

O primeiro pacote a ser compilado será o Binutils porque o configure de execuções de ambos GCC e Glibc vão realizar vários testes de recurso no assembler e linker para determinar quais recursos de software serão ativados e desativados.

Então foi verificado se a variável de ambiente está configurada corretamente, com o comando :

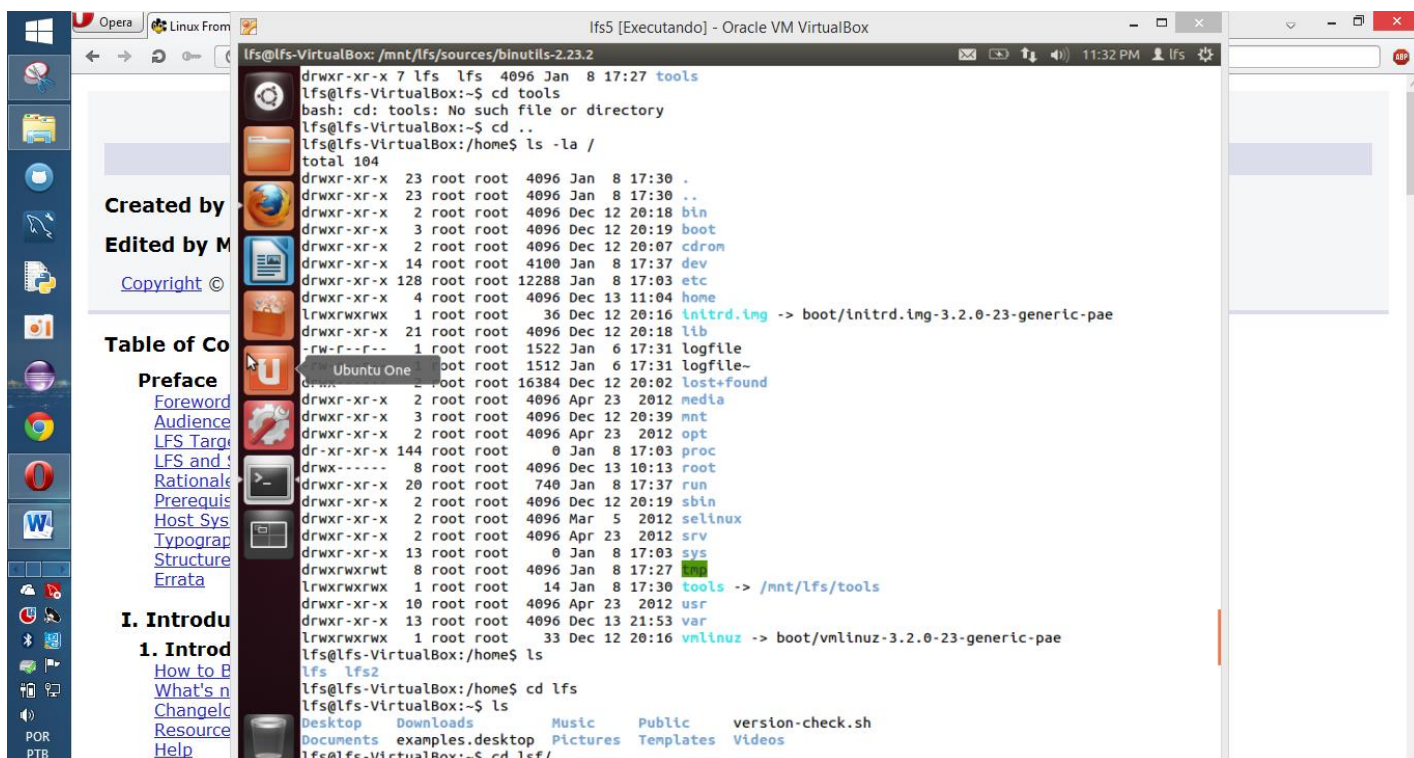
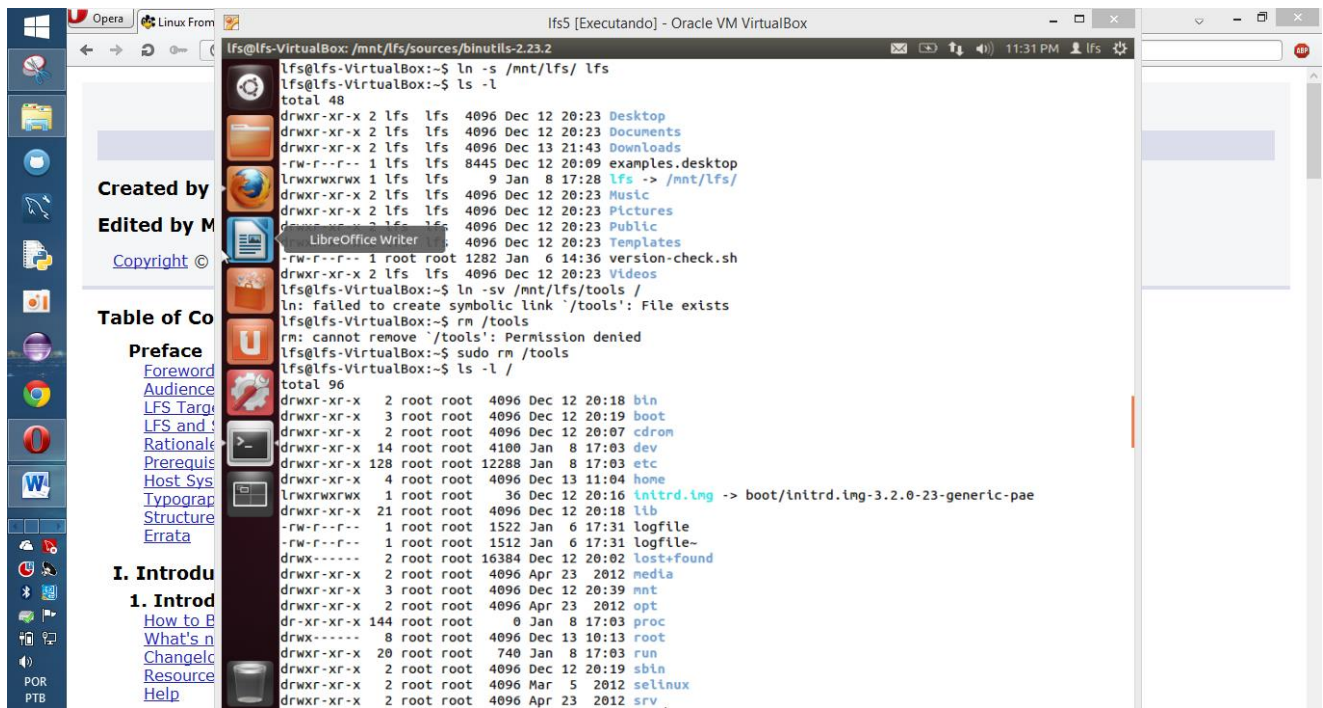
```
Echo $ LFS
```

E o resultado foi: /mnt/lfs

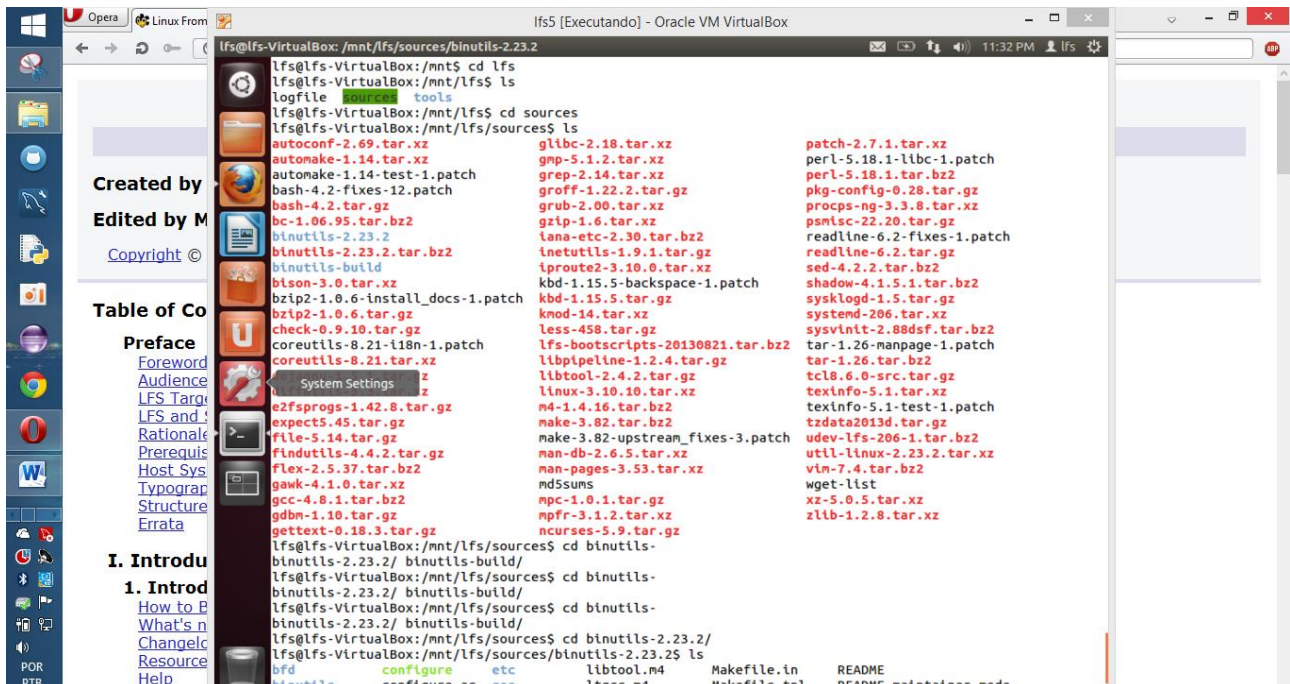
Segundo passo importante foi incluir os links simbólicos corretamente,

```
/usr/bin/awk é um link simbólico para gawk
```

```
/usr/bin/yacc é um link simbólico para bison
```



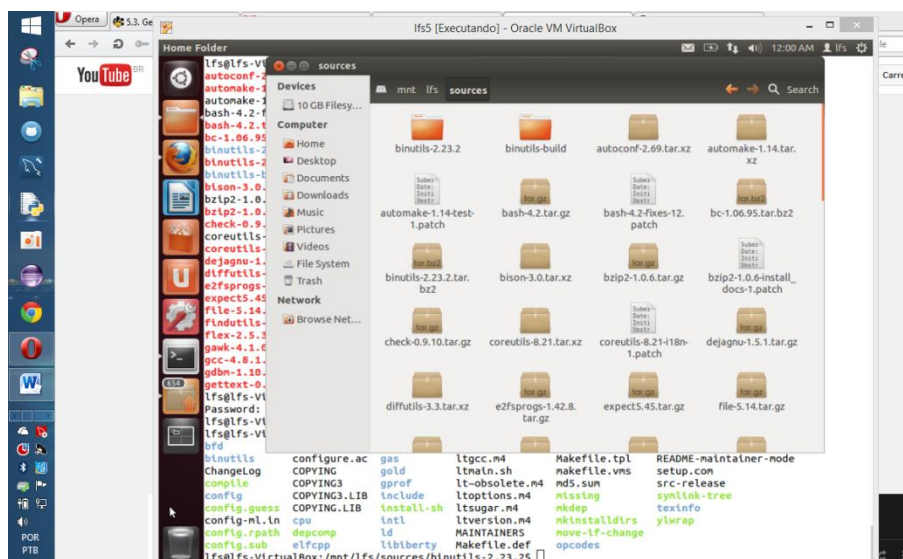
Outro passo proposto no Linux From Scratch foi colocar todas os sources e patches em um diretório que será acessível a partir do ambiente chroot, ou seja colocar tudo em /mnt/LFS/sources / e não em /mnt/LFS/tools/.



Na compilação do Binutils 2.24 foi observado tuas informações importantes tais como : o tempo de compilação aproximada que é de 1SBU e o espaço em disco necessário 404 MB, importante entender essas informações para já ir entendendo os tamanhos dos pacotes para saber o que vamos realmente precisar no LFS final onde teremos que ter um Sistema Operacional de 512 MB.

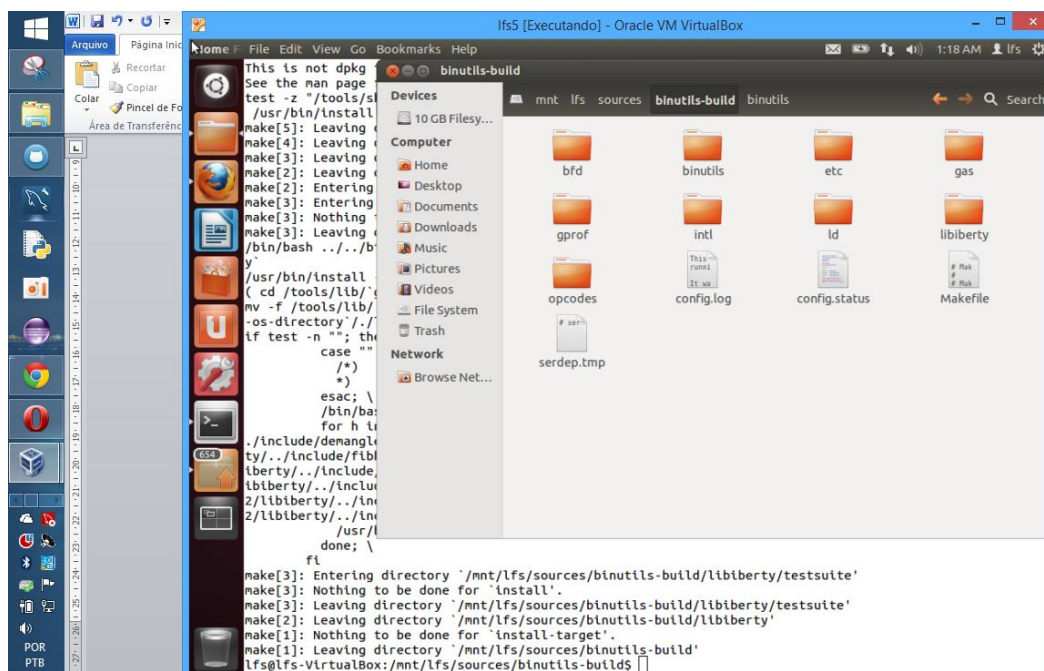
O primeiro passo para compilar o Binutils foi extrair o pacote, onde extrai manualmente na pasta e por comando crei um diretório de compilação dedicado, através do comando :

mkdir -v ../binutils-build

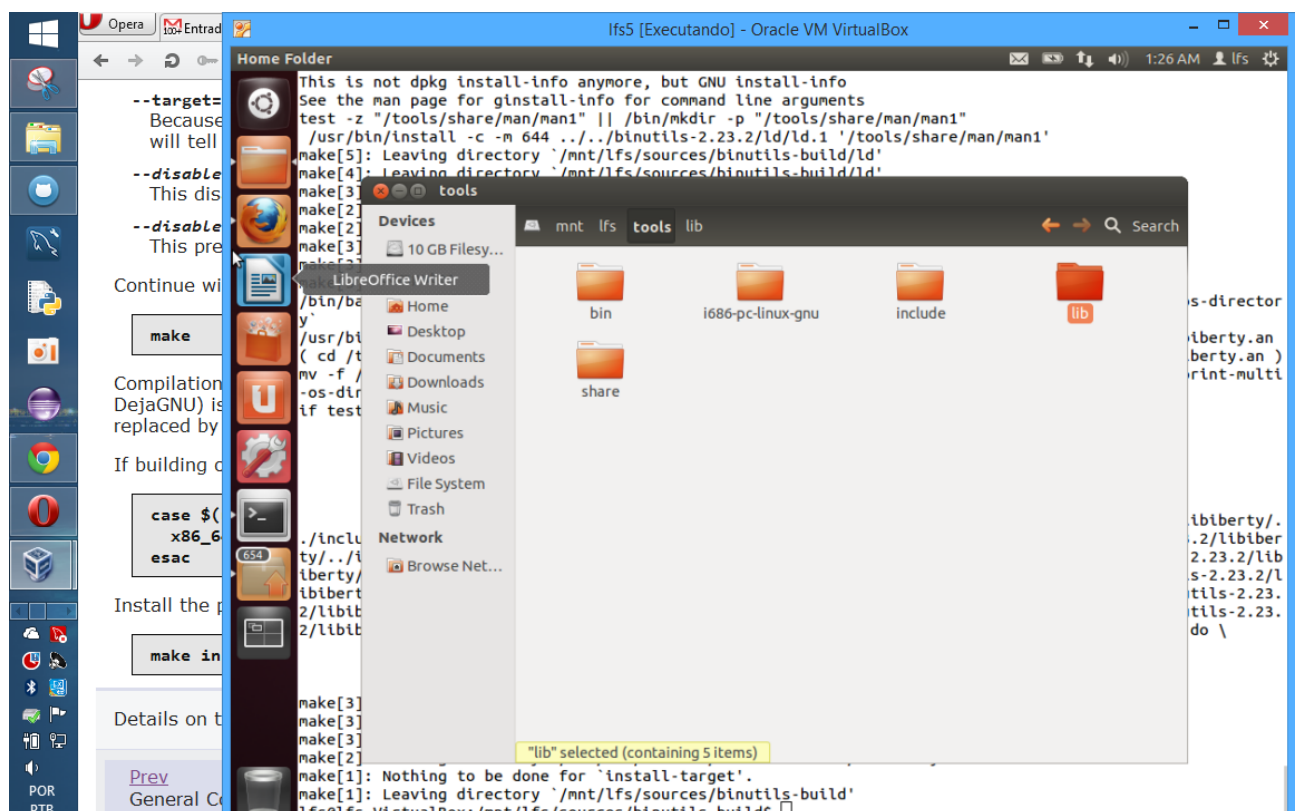


Depois foi dado o comando make install para a instalação do pacote. As duas

figuras a baixo mostra como ficou depois do make install e foi observado que ocorreu mudanças nas pastas Sources e Tools , arquivos novos foram gerados.

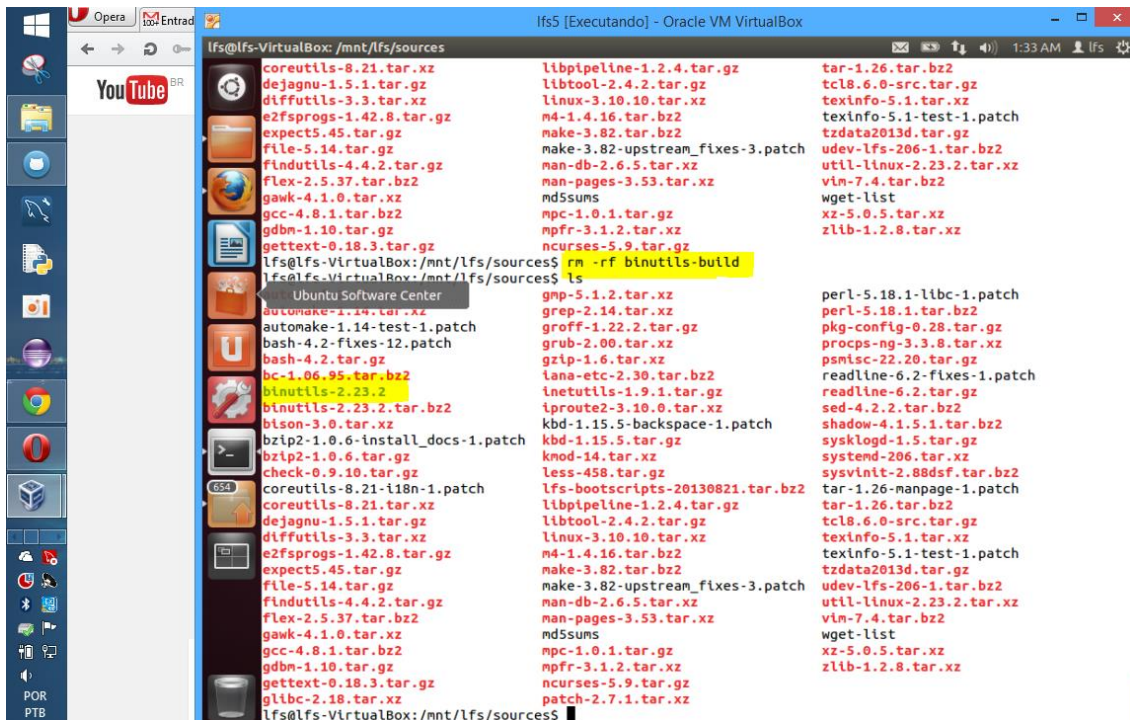


Pasta SOURCES



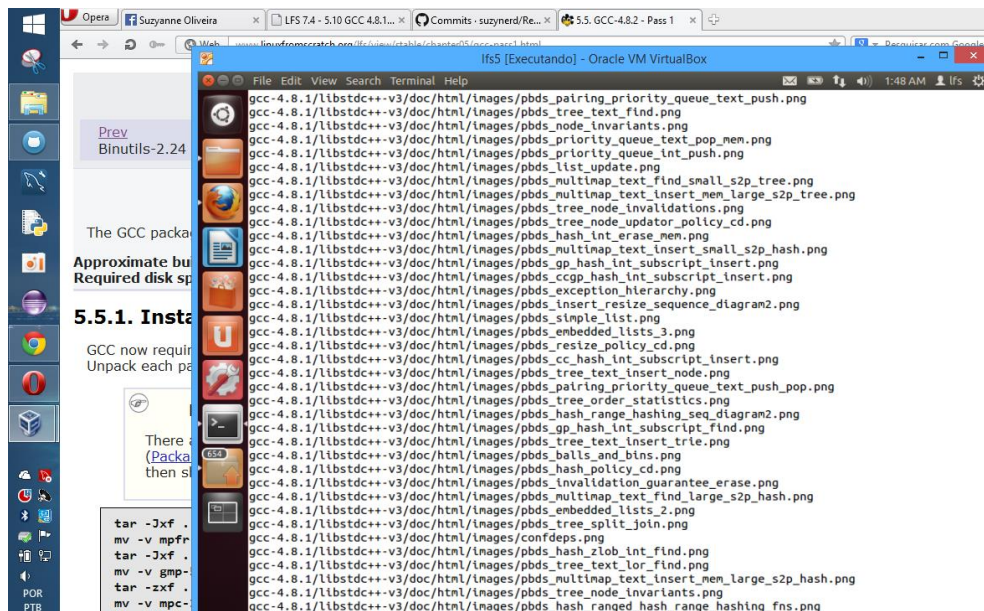
Pasta TOOLS

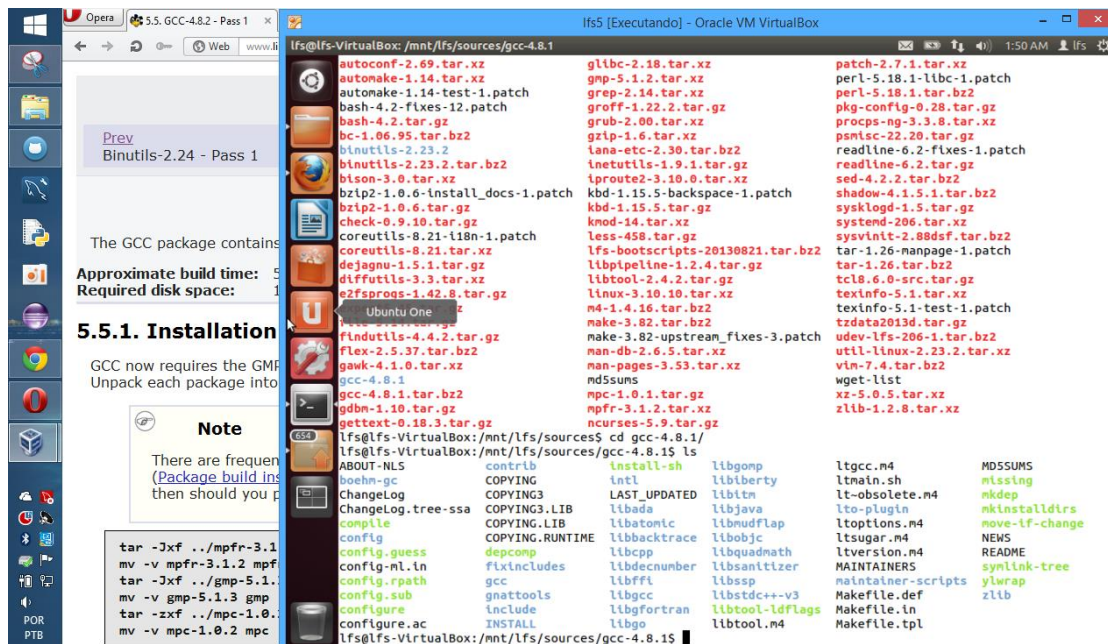
O último passo referente ao Binutils foi remover o pacote que foi extraído [Binutils – Build] com o comando `rm -rf binutils-build` , a figura a baixo mostra com detalhes.



Próxima compilação GCC 4.8.2

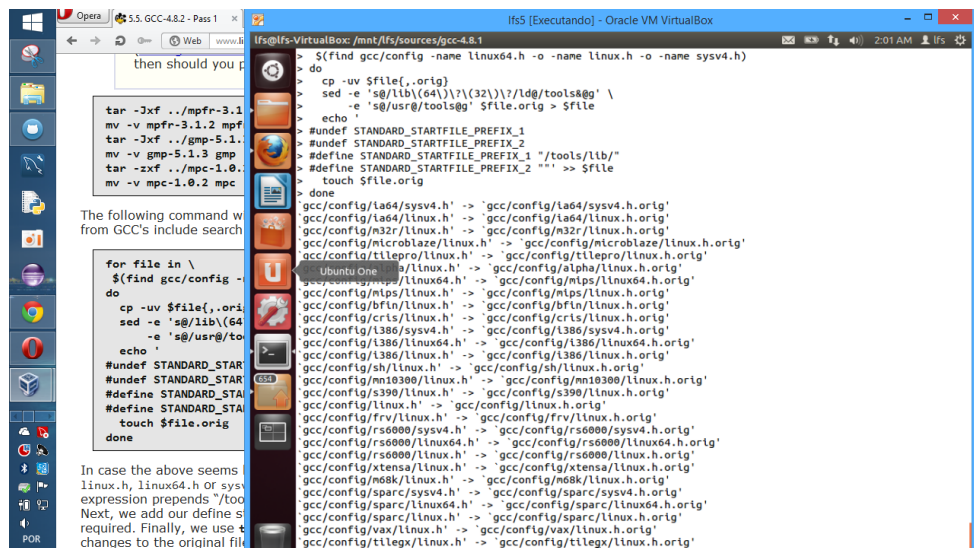
Antes de tudo foi extraído o pacote dentro de sources e cd dentro do diretório para assim começar a configurar para a compilação.





Depois preciso alterar a localização do vinculador dinâmico padrão do GCC para o instalado em /tools.

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



Para não ter problemas na compilação do próximo pacote na compilação do GCC recomenda-se o consertar o problema de proteção de pilha que o gcc não consegue detectar com o seguinte comando :

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

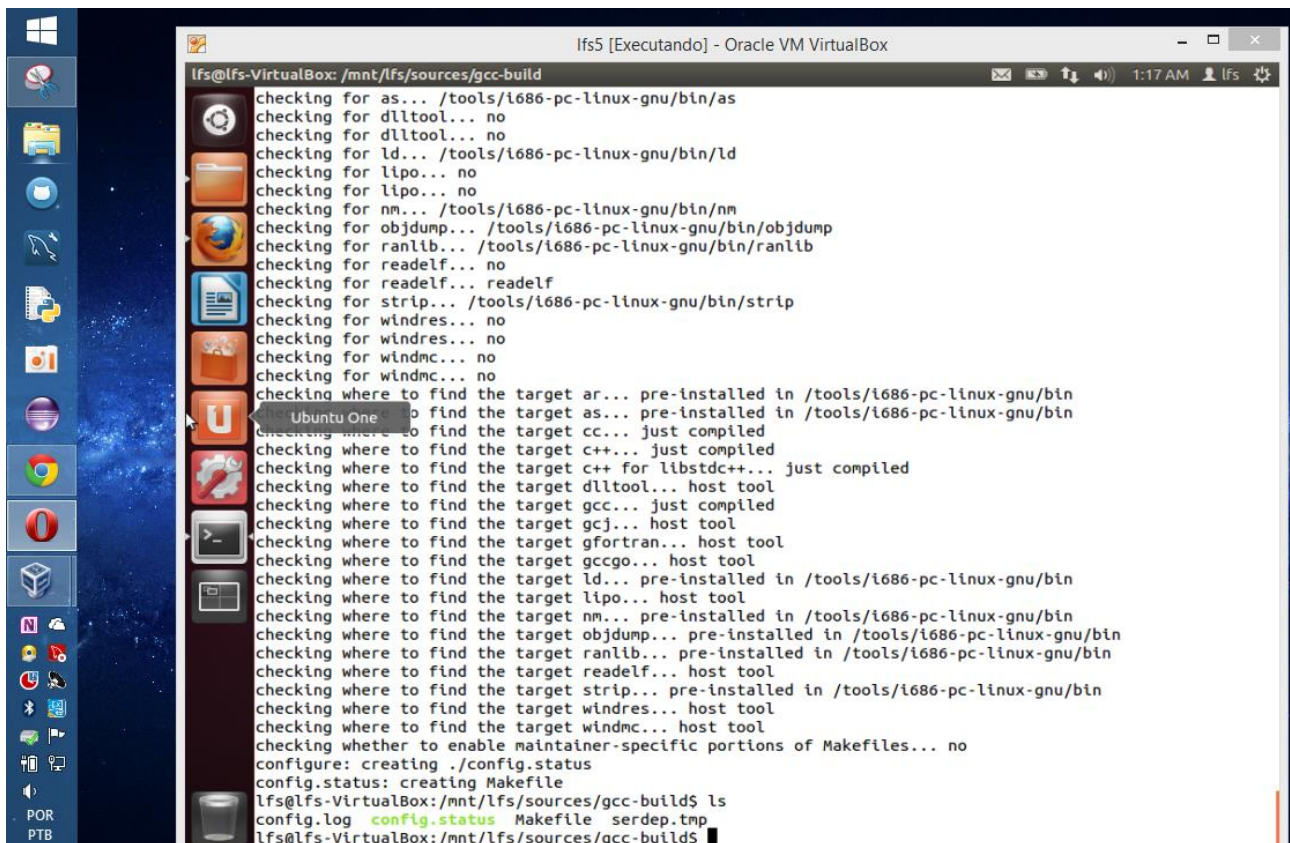
```

lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-4.8.1
> cp -uv $file{,.orig}
> sed -e 's@/lib/(64)?/(32)?/ld@/tools@g' \
  -e 's@/usr/@/tools@g' $file.orig > $file
> echo '
> #undef STANDARD_STARTFILE_PREFIX_1
> #undef STANDARD_STARTFILE_PREFIX_2
> #define STANDARD_STARTFILE_PREFIX_1 "/tools/lib/"
> #define STANDARD_STARTFILE_PREFIX_2 "" >> $file
> touch $file.orig
> done
gcc/config/ia64/sysv4.h' -> 'gcc/config/ia64/sysv4.h.orig'
gcc/config/ia64/linux.h' -> 'gcc/config/ia64/linux.h.orig'
gcc/config/m32r/linux.h' -> 'gcc/config/m32r/linux.h.orig'
gcc/config/microblaze/linux.h' -> 'gcc/config/microblaze/linux.h.orig'
gcc/config/tilepro/linux.h' -> 'gcc/config/tilepro/linux.h.orig'
gcc/config/alpha/linux.h' -> 'gcc/config/alpha/linux.h.orig'
gcc/config/mips/linux64.h' -> 'gcc/config/mips/linux64.h.orig'
gcc/config/mips/linux.h' -> 'gcc/config/mips/linux.h.orig'
gcc/config/bfin/linux.h' -> 'gcc/config/bfin/linux.h.orig'
gcc/config/cris/linux.h' -> 'gcc/config/cris/linux.h.orig'
gcc/config/i386/sysv4.h' -> 'gcc/config/i386/sysv4.h.orig'
gcc/config/i386/linux64.h' -> 'gcc/config/i386/linux64.h.orig'
gcc/config/i386/linux.h' -> 'gcc/config/i386/linux.h.orig'
gcc/config/sh/linux.h' -> 'gcc/config/sh/linux.h.orig'
gcc/config/mn10300/linux.h' -> 'gcc/config/mn10300/linux.h.orig'
gcc/config/s390/linux.h' -> 'gcc/config/s390/linux.h.orig'
gcc/config/linux.h' -> 'gcc/config/linux.h.orig'
gcc/config/frv/linux.h' -> 'gcc/config/frv/linux.h.orig'
gcc/config/rs6000/sysv4.h' -> 'gcc/config/rs6000/sysv4.h.orig'
gcc/config/rs6000/linux64.h' -> 'gcc/config/rs6000/linux64.h.orig'
gcc/config/rs6000/linux.h' -> 'gcc/config/rs6000/linux.h.orig'
gcc/config/xtensa/linux.h' -> 'gcc/config/xtensa/linux.h.orig'
gcc/config/m68k/linux.h' -> 'gcc/config/m68k/linux.h.orig'
gcc/config/sparc/sysv4.h' -> 'gcc/config/sparc/sysv4.h.orig'
gcc/config/sparc/linux64.h' -> 'gcc/config/sparc/linux64.h.orig'
gcc/config/sparc/linux.h' -> 'gcc/config/sparc/linux.h.orig'
gcc/config/vax/linux.h' -> 'gcc/config/vax/linux.h.orig'
gcc/config/tilegx/linux.h' -> 'gcc/config/tilegx/linux.h.orig'
lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-4.8.1$ sed -i '/k prot/agcc_cv_libc_provides_ssp=yes' gcc/configur
e
lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-4.8.1$
  
```

Preparando o GCC para a compilação, comandos rodados perfeitamente gerando o resultado esperado.

```

lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-4.8.1$ ls
ABOUT-NLS      contrib         install-sh      libgomp         ltgcc.m4        MDSSUMS
boehm-gc        COPYING        intl            libiberty       ltmain.sh       missing
ChangeLog       COPYING3       LAST_UPDATED   libitm           lt-obsoleted.m4 akdep
config           COPYING3.LIB   libada         libjava         lto-plugin      akinstalldirs
config.guess     COPYING.LIB    libatomic       libmudflap      ltoptions.m4    move-if-change
config.ml.in     COPYING.RUNTIME libbacktrace   libobjc         ltsugar.m4      mpfr
config.rpath     gcc            libbcp         libquadmath     ltversion.m4    NEWS
config.sub       gnattools     libffi         libsantizer     MAINTAINERS     README
configure        include        libgcc         libstdc++-v3    maintainer-scripts synlink-tree
configure.ac     INSTALL       libgfortran    libtool-ldflags Makefile.def     yllwrap
lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-4.8.1$ ./gcc-4.8.2/configure
--target=$LFS_TGT
--prefix=/tools
--with-sysroot=$LFS
--with-newlib
--without-headers
--with-local-prefix=/tools
--with-native-system-header-dir=/tools/include
--disable-nls
--disable-shared
--disable-multilib
--disable-decimal-float
--disable-threads
--disable-libatomic
--disable-libgomp
--disable-libitm
--disable-libmudflap
--disable-libquadmath
--disable-lsanitizer
--disable-libssp
--disable-libstdc++-v3
--enable-languages=c,c++
--with-mpfr-include=$(pwd)/../gcc-4.8.2/mpfr/src
--with-mpfr-lib=$(pwd)/mpfr/src/.libs
  
```

```
lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-build
checking for as... /tools/i686-pc-linux-gnu/bin/as
checking for dlltool... no
checking for dlltool... no
checking for ld... /tools/i686-pc-linux-gnu/bin/ld
checking for lipo... no
checking for lipo... no
checking for nm... /tools/i686-pc-linux-gnu/bin/nm
checking for objdump... /tools/i686-pc-linux-gnu/bin/objdump
checking for ranlib... /tools/i686-pc-linux-gnu/bin/ranlib
checking for readelf... no
checking for readelf... readelf
checking for strip... /tools/i686-pc-linux-gnu/bin/strip
checking for windres... no
checking for windres... no
checking for windmc... no
checking for windmc... no
checking where to find the target ar... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target as... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target cc... just compiled
checking where to find the target c++... just compiled
checking where to find the target c++ for libstdc++... just compiled
checking where to find the target dlltool... host tool
checking where to find the target gcc... just compiled
checking where to find the target gcj... host tool
checking where to find the target gfortran... host tool
checking where to find the target gccgo... host tool
checking where to find the target ld... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target lipo... host tool
checking where to find the target nm... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target objdump... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target ranlib... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target readelf... host tool
checking where to find the target strip... pre-installed in /tools/i686-pc-linux-gnu/bin
checking where to find the target windres... host tool
checking where to find the target windmc... host tool
checking whether to enable maintainer-specific portions of Makefiles... no
configure: creating ./config.status
config.status: creating Makefile
lfs@lfs-VirtualBox:/mnt/lfs/sources/gcc-build$ ls
config.log config.status Makefile serdep.tmp
lfs@lfs-VirtualBox:/mnt/lfs/sources/gcc-build$
```

Tudo correto e o Make aparecendo na pasta do GCC-build

No 5.6 tem a instalação dos cabeçalhos API LINUX, pois o kernel tem que expor uma Application Programming Interface para a biblioteca C do sistema (glibc no LFS) para isso foi utilizado o seguinte comando:

make mrproper

E para extrair os cabeçalhos do Kernel:

make headers_check

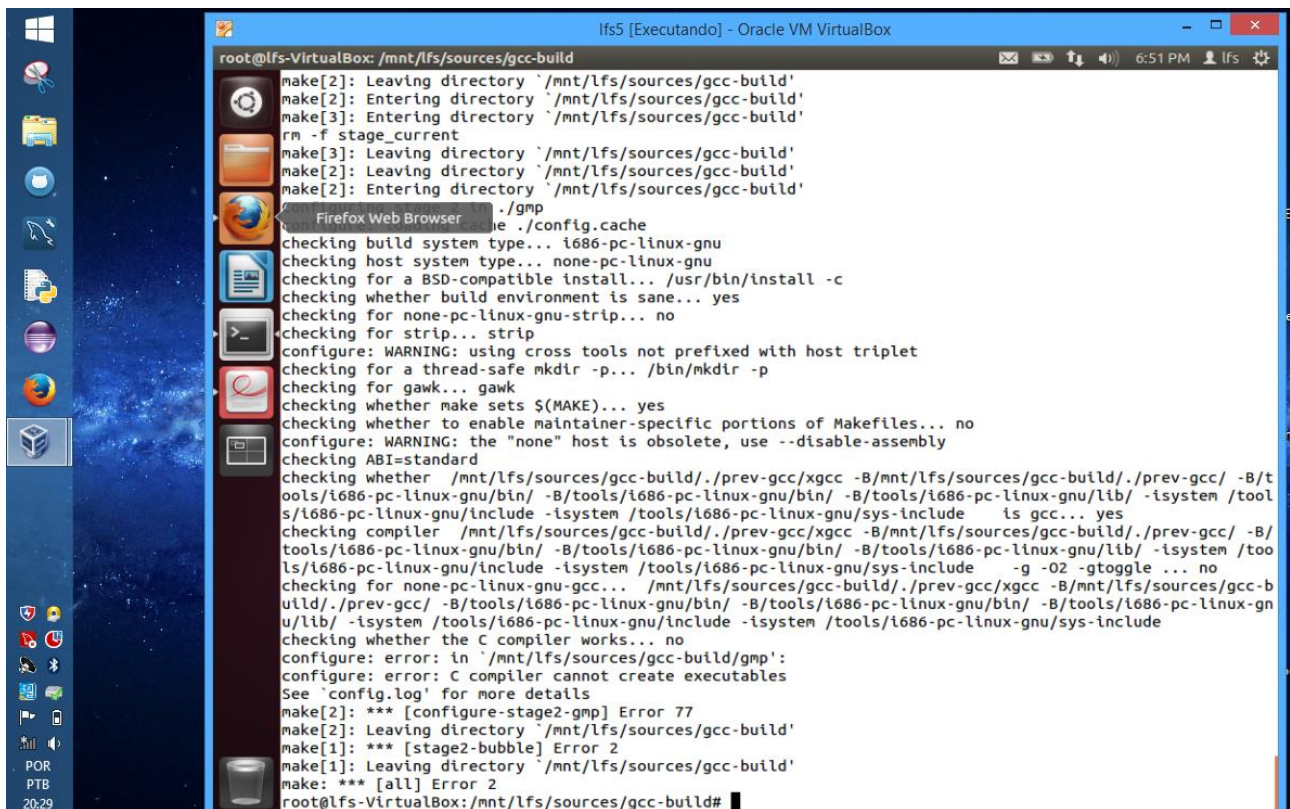
make INSTALL_HDR_PATH=dest headers_install

cp -rv dest/include/* /tools/include

- ***Erro no Make install do GCC***

No comando Make Install deu erro na compilação com isso foi observado que faltava uma variável ser setada \$LFS_TGT ela estava sendo referenciada no comando de configuração para preparação do GCC, tal comando :

```
../gcc-4.8.2/configure \
--target=$LFS_TGT \
--prefix=/tools \
..... \
--with-mpfr-lib=$(pwd)/mpfr/src/.libs
```

```
root@lfs-VirtualBox: /mnt/lfs/sources/gcc-build
make[2]: Leaving directory `/mnt/lfs/sources/gcc-build'
make[2]: Entering directory `/mnt/lfs/sources/gcc-build'
make[3]: Entering directory `/mnt/lfs/sources/gcc-build'
rm -f stage_current
make[3]: Leaving directory `/mnt/lfs/sources/gcc-build'
make[2]: Leaving directory `/mnt/lfs/sources/gcc-build'
make[2]: Entering directory `/mnt/lfs/sources/gcc-build'
./gmp
./configure
checking build system type... i686-pc-linux-gnu
checking host system type... none-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for none-pc-linux-gnu-strip... no
checking for strip... strip
configure: WARNING: using cross tools not prefixed with host triplet
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
configure: WARNING: the "none" host is obsolete, use --disable-assembly
checking ABI=standard
checking whether /mnt/lfs/sources/gcc-build/./prev-gcc/xgcc -B/mnt/lfs/sources/gcc-build/./prev-gcc/ -B/tools/i686-pc-linux-gnu/bin/ -B/tools/i686-pc-linux-gnu/bin/ -B/tools/i686-pc-linux-gnu/lib/ -isystem /tools/i686-pc-linux-gnu/include -isystem /tools/i686-pc-linux-gnu/sys-include is gcc... yes
checking compiler /mnt/lfs/sources/gcc-build/./prev-gcc/xgcc -B/mnt/lfs/sources/gcc-build/./prev-gcc/ -B/tools/i686-pc-linux-gnu/bin/ -B/tools/i686-pc-linux-gnu/bin/ -B/tools/i686-pc-linux-gnu/lib/ -isystem /tools/i686-pc-linux-gnu/include -isystem /tools/i686-pc-linux-gnu/sys-include -g -O2 -gtoggle ... no
checking for none-pc-linux-gnu-gcc... /mnt/lfs/sources/gcc-build/./prev-gcc/xgcc -B/mnt/lfs/sources/gcc-build/./prev-gcc/ -B/tools/i686-pc-linux-gnu/bin/ -B/tools/i686-pc-linux-gnu/bin/ -B/tools/i686-pc-linux-gnu/lib/ -isystem /tools/i686-pc-linux-gnu/include -isystem /tools/i686-pc-linux-gnu/sys-include
checking whether the C compiler works... no
configure: error: in `/mnt/lfs/sources/gcc-build/gmp':
configure: error: C compiler cannot create executables
See `config.log' for more details
make[2]: *** [configure-stage2-gmp] Error 77
make[2]: Leaving directory `/mnt/lfs/sources/gcc-build'
make[1]: *** [stage2-bubble] Error 2
make[1]: Leaving directory `/mnt/lfs/sources/gcc-build'
make: *** [all] Error 2
root@lfs-VirtualBox: /mnt/lfs/sources/gcc-build#
```

Na instalação do GCC houve a compreensão de algumas coisas, tais como:

O que é o Make, makefile e Make install?

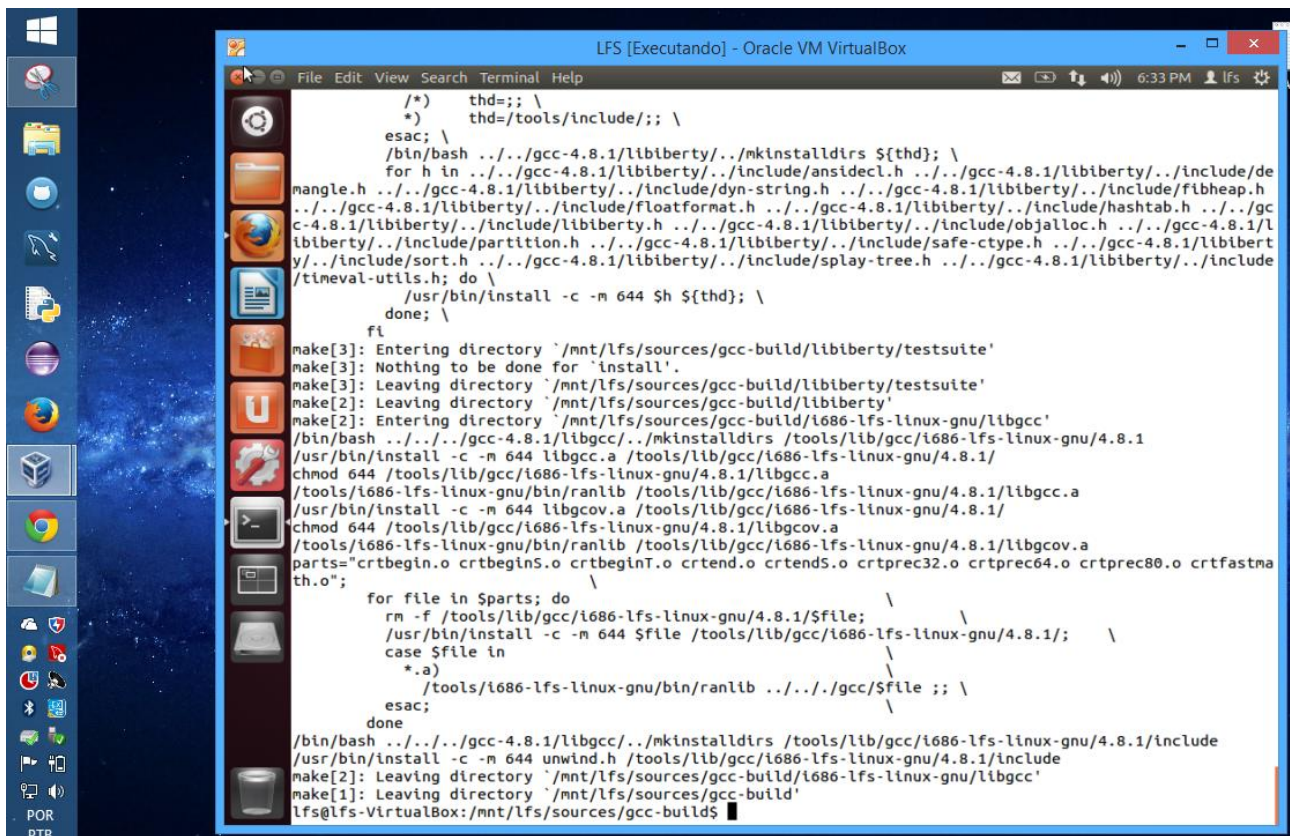
Make é, de forma geral, um automatizador de tarefas: ele possibilita que você crie scripts para tarefas comuns e os nomeie. Assim, quando precisar executar dada tarefa, pode executar apenas *make <nome_da_tarefa>*. Essas tarefas ficam no arquivo Makefile. O comando *make install* executa a tarefa chamada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

GCC COMPILADO COM SUCESSO

O gcc depois foi compilado com sucesso depois das modificações necessárias feitas como foi falado anteriormente. A figura abaixo é a tela de compilação do GCC com sucesso.



```
lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-build$
/*) thd=;; \
*) thd=/tools/include;; \
esac; \
/bin/bash ../../gcc-4.8.1/libiberty/./mkinstalldirs ${thd}; \
for h in ../../gcc-4.8.1/libiberty/./include/ansidecl.h ../../gcc-4.8.1/libiberty/./include/de
mangle.h ../../gcc-4.8.1/libiberty/./include/dyn-string.h ../../gcc-4.8.1/libiberty/./include/fibheap.h
../../gcc-4.8.1/libiberty/./include/floatformat.h ../../gcc-4.8.1/libiberty/./include/hashtab.h ../../gc
c-4.8.1/libiberty/./include/libiberty.h ../../gcc-4.8.1/libiberty/./include/objalloc.h ../../gcc-4.8.1/l
ibiberty/./include/partition.h ../../gcc-4.8.1/libiberty/./include/safe-ctype.h ../../gcc-4.8.1/libibert
y/./include/sort.h ../../gcc-4.8.1/libiberty/./include/splay-tree.h ../../gcc-4.8.1/libiberty/./include
/timeval-utils.h; do \
    /usr/bin/install -c -m 644 $h ${thd}; \
done; \
fi
make[3]: Entering directory `/mnt/lfs/sources/gcc-build/libiberty/testsuite'
make[3]: Nothing to be done for `install'.
make[3]: Leaving directory `/mnt/lfs/sources/gcc-build/libiberty/testsuite'
make[2]: Leaving directory `/mnt/lfs/sources/gcc-build/libiberty'
make[2]: Entering directory `/mnt/lfs/sources/gcc-build/i686-lfs-linux-gnu/libgcc'
/bin/bash ../../gcc-4.8.1/libgcc/./mkinstalldirs /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1
/usr/bin/install -c -m 644 libgcc.a /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/
chmod 644 /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/libgcc.a
/tools/i686-lfs-linux-gnu/bin/ranlib /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/libgcc.a
/usr/bin/install -c -m 644 libgccov.a /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/
chmod 644 /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/libgccov.a
/tools/i686-lfs-linux-gnu/bin/ranlib /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/libgccov.a
parts="crtbegin.o crtbeginS.o crtbeginT.o crtend.o crtendS.o crtendT.o crtfastma
th.o";
for file in $parts; do
    rm -f /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/$file; \
    /usr/bin/install -c -m 644 $file /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/; \
    case $file in
        *.a)
            /tools/i686-lfs-linux-gnu/bin/ranlib ../../gcc/$file ;; \
        *)
            ;;
    esac;
done
/bin/bash ../../gcc-4.8.1/libgcc/./mkinstalldirs /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/include
/usr/bin/install -c -m 644 unwind.h /tools/lib/gcc/i686-lfs-linux-gnu/4.8.1/include
make[2]: Leaving directory `/mnt/lfs/sources/gcc-build/i686-lfs-linux-gnu/libgcc'
make[1]: Leaving directory `/mnt/lfs/sources/gcc-build'
lfs@lfs-VirtualBox: /mnt/lfs/sources/gcc-build$
```

Compilação do GCC com sucesso

Depois do GCC instalado e **funcionando corretamente** consegui prosseguir no projeto na compilação de outros pacotes do sistema temporário LFS, esses pacotes foram:

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

Os pacotes com problemas na compilação:

DejaGNU-1.5.1

Ncurses-5.9

Sed-4.2.2

Texinfo-5.2

OBS : Em anexo na pasta PRINTS encontram-se as imagens de compilação de cada patoce.

Todos os programas que serão compilados no capítulo 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 removê-los após o sistema estar pronto.

➤ **Por isso foi criado o diretório:**

`mkdir -v $LFS/tools`

➤ **E um link simbólico para nosso sistema host:**

`ln -sv $LFS/tools /`

Logado como root é fácil danificar o sistema. Então foi criado um usuário LFS para fazermos o trabalho. Então foi criado de acordo com o livro:

`groupadd lfs`

E o usuário:

`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

Outra coisa importante é fazer o lfs dono do diretório \$LFS/tools para garantir o acesso total à ele:

`chown -v lfs $LFS/tools`

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

`chown -v lfs $LFS/sources`

É importante logar com o novo usuário usando o comando :

`su – lfs`

O sinal “-” instrui o “su” para iniciar um shell de login ao invés do shell sem login.

Configurando o ambiente do usuário lfs

Para um bom ambiente de trabalho criando 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 aqui (HOME, TERM e PS1), garantindo que nenhuma variável indesejada (e potencialmente perigosa) do sistema host atrapalhe no ambiente de desenvolvimento.

A nossa atual instância de shell é uma instância sem login, então ela não lê o `/etc/profile` nem `.bash_profile`, mas lê o `.bashrc`. Então 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
```

O que acontece:

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. Porém, as novas ferramentas devem ser usadas assim que instaladas, então o **bash** sempre vai procurar no **PATH** ao invés de tentar executar a última, encontrando assim sempre a mais nova compilada.

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 o ponto de montagem.

A variável `LC_ALL` controla algumas configurações de alguns programas, relacionadas a localização (país), seguindo certas convenções. Se o sistema host usa a Glibc anterior a 2.2.4, 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.

E então executamos o profile:

```
source ~/.bash_profile
```

Sobre o tempo de compilação

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. Em <http://www.linuxfromscratch.org/~sbu/> teremos uma ideia dos sistemas e dos tempos de cada um.

Conclusão

O projeto foi excelente no ponto de vista de aprendizagem sobre como um sistema operacional funciona e conhecimento de comandos de shell. Vários erros durante o processo de construção prejudicaram o tempo para avançar no projeto nos outros capítulos, inicialmente foi muito complicado entender o que estávamos fazendo e como seria feito e diante de problemas na execução de scripts, na hora de particionar a máquina virtual onde fizemos errado e perdemos tudo o que fizemos isso tudo gerou danos em questão de tempos e prazos.

O primeiro capítulo do LFS descreve como será o processo de construção o capítulo 2 descreve a criação de um sistema de arquivos nativo e partições do Linux, o capítulo 3 descreve os pacotes e patches que serão necessários para download e com armazená-los no novo sistema de arquivos, o capítulo 4 descreve como configurar um novo ambiente funcional, explicando vários enganos normais a serem evitados e o capítulo 5 explica a instalação de pacotes que formarão a suite de desenvolvimento básica (chamada de toolchain) que será usada para a construção do novo sistema, construindo o primeiro passo que inclui o Binutils e o GCC, indo para o segundo passo criando a Glib e linkando dinamicamente a nova toolchain para a nova Glib. Ao final da Parte 5, o processo de instalação do LFS não dependerá mais do Linux anterior (host), com exceção do kernel que está rodando. Até o desenvolvimento básico do sistema foram vários dias e horas tentando concertar erros de compilação de pacotes visto que alguns deles chegaram há 20 minutos e às vezes dava problema e tinha que voltar o processo tudo de novo tudo isso lembrando que existia uma forte dependência entre os pacotes se um desse problema tinha que concertar para passar para o próximo, diante aos fatos o desenvolvimento parava até que tivesse sucesso nessa etapa.

Faltou finalizar o capítulo 6 com onde o sistema completo será montado usando o programa chroot para entrar em um ambiente virtual, usando um shell em que o diretório root será setado à partição do LFS. É como reiniciar a máquina e instruir o kernel a montar e utilizar a partição do LFS ao invés do sistema original. Isso por que não é necessário todo o trabalho de criar todo o ambiente de boot neste momento, o capítulo 7 que é configurar um bootscript e o capítulo 8 que mostra como configurar o kernel e o bootloader.

