Universidad del Valle de Guatemala
Sistemas Operativos
Catedrático: Tomás Gálvez
Semestre II, 2018

Proyecto de semestre: LFS

Fecha de Entrega: última semana de clases.

Descripción: este proyecto reta al estudiante a armar (*put together*) su propio sistema operativo Linux siguiendo la guía del proyecto *Linux From Scratch* (http://www.linuxfromscratch.org/). A lo largo del desarrollo del proyecto se plantean muchas preguntas con cuya respuesta el estudiante profundizará sus conocimientos sobre los componentes esenciales y opcionales de un sistema operativo, así como su relación y experiencia con Linux. Al final del semestre, el estudiante deberá presentar su LFS funcionando en una máquina virtual o computadora física, así como un documento bien organizado que presente las respuestas a las preguntas planteadas en este proyecto.

Materiales: se requerirá el software VirtualBox (https://www.virtualbox.org/) y el LiveCD en inglés del sistema operativo Knoppix (http://torrent.unix-ag.uni-kl.de/). De Knoppix se necesita la versión 7.2 de 32 bits; se identifica por su tamaño de 701.07 MB (cuide descargar la versión en inglés [EN]). Knoppix se descarga por medio de BitTorrent, por lo que necesitará software para descargarlo (se recomienda µTorrent). Probablemente también necesite software para montar el LiveCD de Knoppix en su computadora. De necesitarlo, se recomienda Virtual CloneDrive (https://www.elby.ch/en/products/vcd.html).

Nota: tiene la libertad de desarrollar su LFS en el sabor de Linux de su preferencia. En el curso se ha usado previamente la versión indicada de Knoppix, con las siguientes ventajas: ha funcionado y no es muy grande, por lo que se descarga rápido y ocupa poco espacio. Por tanto, este documento supone que usted usa este sistema operativo como base. Sin embargo, es una versión antigua de un sistema operativo con escasa documentación en línea. Si usted decide usar otro sistema operativo (o incluso otra versión de Knoppix), por favor tome en cuenta lo siguiente:

- Es probable que se enfrente a complicaciones adicionales con el proyecto, pero también es probable que le resulte más sencillo, en especial si usa una distribución popular de Linux.
- Debe responder a las preguntas en este documento sin importar qué guía siga ni qué sistema operativo use. Las únicas excepciones serán las preguntas que estén exclusivamente relacionadas con el sistema Knoppix, si las hubiere. Se recomienda consultar en caso de duda.
- Por la orientación de este documento, quizá le resulte más conveniente seguir la guía oficial de *Linux From Scratch*, descargable desde el *link* en la descripción. Se recomienda seguir la versión 8.1, que es el fundamento de esta guía. A la fecha de creación de esta guía, la versión más reciente es la 8.2. También se aceptará su proyecto si es desarrollado con esta versión.

Aunque use la versión de Knoppix recomendada en este documento, puede seguir la guía oficial en sus versiones 8.1 y 8.2 para armar su LFS. Naturalmente, se recomienda seguir una única guía, y tendrá que responder a las preguntas de este documento sin importar cuál guía use.

IMPORTANTE: sin importar el sistema operativo que use como base, se recomienda trabajar el proyecto en una máquina virtual y crear muchos *snapshots*. El proyecto le exigirá realizar operaciones que pueden poner en riesgo el correcto funcionamiento de su computadora. Cada estudiante será únic@ responsable de lo que suceda con el equipo sobre el que trabaje.

Contenido

Preguntas

Preparación de la máquina virtual con Knoppix

a. Abra VirtualBox y presione *New* para crear una máquina virtual nueva.
b. Póngale un nombre descriptivo, seleccione Linux como sistema operativo y Linux 2.6 de 32 bits como versión.
c. Dé *Next* cuando se le pregunte cuánta RAM proveer a su máquina virtual para usar la cantidad recomendada, aunque se tendrá mejor desempeño si se asignan al menos 2048 MB de RAM.
d. En la creación de disco duro seleccione *Create a virtual hard drive now*. Cuando se le pregunte el tipo seleccione VDI y cuando se le pregunte el método de reserva de espacio seleccione *Dynamically allocated*. Finalmente, seleccione un tamaño de 20 GB de disco duro y dé *click* en *Create.*
e. Monte el LiveCD de Knoppix (si usa Virtual CloneDrive, *click* derecho al archivo y luego *Mount*).
f. Inicie la máquina virtual (si se le pregunta con qué disco iniciar el sistema, seleccione el LiveCD).
g. Deberá aparecer una pantalla con un "pingüino de Vitruvio" donde debe presionar *Enter*; esto iniciará el sistema operativo.
h. Inicie un *terminal emulator* (ubicado en la barra de inicio) y en ella el programa `cfdisk`.
i. Cree cuatro particiones, todas primarias y, en cada ocasión, desde *beginning*. La primera debe ser de 100 MB, la segunda de 10,000 MB, la tercera de 2,000 MB y la última del tamaño sugerido por la herramienta. Sólo la última debe ser *bootable*. No olvide seleccionar la opción *Write* para que sus cambios se graben en el disco virtual.
j. Seleccione la partición de 2,000 MB y vaya a la opción *Type*. Allí dé *Enter* y luego ingrese el tipo 82. El tipo deberá cambiar a *Linux swap.*
k. Reinicie el sistema (es probable que al iniciar la máquina virtual deba volver a montar el LiveCD).
l. Vuelva a ejecutar `cfdisk` en el *terminal emulator* y verifique el nombre de las últimas dos particiones (probablemente sean `sda3` y `sda4`).
m. Salga de `cfdisk` y ejecute los siguientes comandos (con los nombres verificados en el paso anterior):

```
sudo mkreiserfs /dev/sda4
sudo mkswap /dev/sda3
```

Apruebe cualquier verificación que le pida el sistema.

n. Cierre la terminal y dé *click* en el ícono triangular que se encuentra en la esquina inferior izquierda. Allí seleccione el sub-menú Knoppix y dé *click* en la herramienta Knoppix HD Install.
o. Se le preguntará dos veces si desea continuar. Dele que sí. Luego deberá aparecer un menú con varias opciones, la primera de las cuales debe ser `/dev/sda4` (o el nombre correspondiente a la última partición creada). Si este es el caso, seleccione dicha opción y continúe. Si no aparece la opción debe haber cometido un error. Borre las particiones y vuelva a intentar desde el paso *e*. Si aun así no aparece la opción, borre la máquina virtual y repita todos los pasos.
p. Al finalizar la instalación se mostrará un diálogo explicando cómo instalar GRUB. Luego de hacer *click* en OK se preguntará si se desea instalar el *bootstrap loader* GRUB. Seleccione la opción que

sí instala GRUB y continúe. Luego de unos momentos se mostrará un diálogo indicando la exitosa instalación de Knoppix y se pedirá que se reinicie el sistema.

q. Reinicie el sistema sin volver a montar el LiveCD. Si todo ha salido bien, debería mostrarse un menú de color azul cuya única opción dice KNOPPIX. Al dar *Enter* se deberá iniciar el sistema operativo desde el disco duro. Notará la diferencia contra el sistema iniciado desde el LiveCD por la falta de íconos en el escritorio.

Instalación de herramientas necesarias para la creación de un LFS

En el proceso se obtendrá (o reforzará) el conocimiento de comandos básicos del *shell* de Linux, así como el procedimiento de instalación de una *tarball*. También se usará el manejador de paquetes `apt-get`.

a. Encienda su máquina virtual con Knoppix e inicie una terminal.
b. Diríjase hacia su directorio base y cree allí un directorio llamado `lfsreqs` usando los siguientes comandos:

```
cd ~
sudo mkdir lfsreqs
```

c. Copie al directorio recién creado el *script* de *shell code* para revisión de requerimientos llamado `version-check.sh`, provisto en Canvas. Revise el código para responder lo siguiente:

- ¿Por qué las condiciones de los *if*'s están entre corchetes?
- ¿Cuáles son las condiciones que se están verificando en cada caso?

d. Descargue de Canvas el otro archivo de *Linux shell code* llamado `library-check.sh`. Revíselo para responder lo siguiente:

- ¿Para qué sirve encerrar una lista de *strings* entre llaves, en este caso?
- ¿Para qué sirve '$' en *shell code*?
- ¿Qué se está haciendo con este ciclo *for*?

e. Ejecute el primer *script* usando el siguiente comando:

```
sudo bash version-check.sh
```

Habrá algunas herramientas que se reporten no instaladas o mal configuradas.

f. Ejecute el segundo *script* de forma similar. El resultado debe ser tres *not found* o tres *found*. No debe haber una mezcla de *found*'s y *not found*'s.

Instalación de `m4` y `bison`

g. En este paso se instalarán dos herramientas de la manera estándar en Linux. No es necesario realizar este paso de esta forma, pero se hará así como preparación para operaciones futuras. Se recomienda tomar un *snapshot* de la máquina virtual antes de continuar.
Descargue la herramienta `m4` usando `wget`. La dirección es http://ftp.gnu.org/gnu/m4/, y debe descargar a versión 1.4.10. Acceda al *link* para ver el nombre de la *tarball* necesaria. De la misma manera descargue la versión 2.3 de `bison`. La dirección es http://ftp.gnu.org/gnu/bison/.

- ¿Qué es una *tarball*?
- ¿Por qué las *tarball* tienen una extensión "doble" (`.tar.gz`)?

A continuación, deberá instalar `m4`. Para hacerlo comience por extraer `m4` usando el siguiente comando:

```
tar -xvzf m4-1.4.10.tar.gz
```

- ¿Qué significa cada una de las opciones (`-xvzf`) utilizadas?
- ¿Cuál de estas opciones podría haberse omitido?

Al extraer el paquete encontrará un directorio llamado `m4` (para verlo despliegue el contenido del directorio actual usando el comando `ls`). Acceda al directorio `m4` y lea el archivo `INSTALL` usando los programas `cat`, `nano`, o `less`.

- ¿Cuáles son los tres comandos que se deben ejecutar para instalar un paquete en Linux?
- ¿Dónde encontramos información adicional de instalación específica para cada paquete?

`m4` no tiene requerimientos especiales para nuestros propósitos, entonces proceda a instalarlo usando las instrucciones encontradas en `INSTALL`.

Habiendo instalado `m4` exitosamente repita el proceso de extracción e instalación con `bison`.

h. Desde su directorio base ejecute nuevamente el *script* `version-check.sh`. Notará que el primer error encontrado es "`yacc not found`". Para componer este problema diríjase al directorio `/usr/local/bin` y compruebe la presencia de `bison` en este directorio usando `ls`. Si está, cree un enlace simbólico a este programa llamado `yacc`, ubicado en `/usr/bin`. Para hacerlo ejecute el siguiente comando:

```
sudo ln -s /usr/local/bin/bison /usr/bin/yacc
```

- ¿Qué es un enlace simbólico?
- ¿Cuál es la diferencia entre un enlace simbólico (*symlink*) y uno duro (*hard link*)?
- En el comando `ln` usado para crear enlaces, ¿en qué orden se deben escribir los parámetros para crear un enlace de *A* a *B*?

i. Vuelva a ejecutar `version-check.sh` y compruebe que todo esté bien con `bison` y `yacc`.

Instalación de herramientas faltantes usando la Advanced Package Tool (apt)

j. Ahora deberá instalar las herramientas que hagan falta. Para ello usará el comando `apt-get`. Comience por editar el archivo `/etc/apt/sources.list`. Comente todas las líneas exceptuando las que empiezan con `http://security.debian.org` y la que empieza con `http://debian-knoppix.alioth.debian.org`. Al final del archivo agregue la siguiente línea:

```
deb http://ftp.us.debian.org/debian wheezy main
```

k. Se recomienda tomar un *snapshot* de la máquina antes de ejecutar las instrucciones a continuación. En la terminal actualice `apt-get` con el siguiente comando:

```
sudo apt-get update
```

Cuando este proceso hubiere terminado instalará los paquetes faltantes usando el siguiente comando (variando el nombre de paquete):

```
sudo apt-get install nombredepaquete=version
```

La versión no es obligatoria. Si se omite "`=version`" se instalará la versión más reciente disponible.

Hasta este punto, el resultado de `version-check.sh` debería indicar que sólo faltan los programas `makeinfo` y `gcc`.

- ¿Qué es `apt-get` y cómo funciona?
- ¿Qué se modificó y se hizo con `apt-get` mediante las instrucciones anteriores?

l. El comando `makeinfo` se hace válido con la instalación de un paquete llamado `texinfo`. Se requiere como mínimo la versión 4.7.

m. Proceda a instalar `g++` usando `apt-get`. Debe instalarse la misma versión que tenga para GCC (puede verificarla usando el comando `gcc --version`), aunque esto debería suceder de forma automática.

n. Ejecute nuevamente `version-check.sh` y compruebe que todo está bien. Compruebe también que las versiones de las herramientas son como mínimo las siguientes:

a. `Bash-3.2`
b. `Binutils-2.17` (no mayor a versión 2.27)
c. `Bison-2.3`
d. `Bzip2-1.0.4`
e. `Coreutils-6.9`
f. `Diffutils-2.8.1`
g. `Findutils-4.2.31`
h. `Gawk-4.0.1`
i. `GCC-4.7` (no mayor a versión 6.3.0)
j. `g++ 4.7` (no mayor a versión 6.3.0)

k. `Glibc-2.11` (no mayor a versión 2.25)
l. `Grep-2.5.1a`
m. `Gzip-1.3.12`
n. `Linux Kernel-2.6.32`
o. `M4-1.4.10`
p. `Make-3.81`
q. `Patch-2.5.4`
r. `Perl-5.8.8`
s. `Sed-4.1.5`
t. `Tar-1.22`
u. `Texinfo-4.7`
v. `Xz-5.0.0`

Si todo ha salido bien hasta ahora, cree un *snapshot* de su máquina virtual.

o. Ubique en su computadora física el programa `VBoxManage`. En sistemas Windows debería estar en "*Program files*" o "Archivos de Programa", bajo la carpeta "Oracle".
p. Una vez encontrado el programa ejecute una línea de comandos o terminal ubicada en ese directorio.
q. A continuación, *clickee* "*File*" o "Archivo" en su cliente de VirtualBox (no en una máquina virtual en particular) y seleccione la opción *Virtual Media Manager*. Se abrirá una ventana con descripciones sobre cada disco duro usado por las máquinas virtuales que posea. Copie la ubicación del disco duro usado por la máquina donde tenga instalado Knoppix, y ejecute el siguiente comando en la línea de comandos que abrió (usando la dirección del disco que acaba de copiar y sin omitir las comillas):

```
VBoxManage.exe showhdinfo "Dirección/de/disco"
```

Copie o capture el texto resultante de la ejecución de este comando, e inclúyalo en el archivo donde haya respondido las preguntas de este proyecto. Además, incluya entre sus respuestas una captura de pantalla donde se muestre esta información junto con los resultados positivos de `version-check.sh`.

Preparación de un ambiente aislado para creación de un LFS

a. Abra una terminal.
b. Cree un sistema de archivos adecuado para cada una de las particiones del LFS usando el siguiente comando:

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

donde `<tipo>` debe ser `ext2` para la partición `boot` (la de 100 MB) y `ext4` para la partición que contendrá el LFS; y `<xxx>` debe ser la partición correspondiente a cada caso.

c. Diríjase a su directorio `~` y cree un archivo llamado `.profile`. En éste incluya las siguientes líneas:

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

```
sudo mount -v -t ext4 /dev/<xxx> $LFS
sudo mount -v -t ext2 /dev/<yyy> $LFS/boot
```

donde `<xxx>` es la partición donde se creará el LFS y `<yyy>` es la partición `boot`.

- ¿Para qué sirve crear un archivo llamado `.profile`?

d. Cree los directorios donde se montarán las particiones del inciso anterior.
e. Reinicie su sistema.
f. Al iniciar abra nuevamente una terminal y asegúrese de que los comandos en `.profile` se estén ejecutando por medio de las siguientes instrucciones:

```
echo $LFS
```

y

```
sudo mount
```

g. Cree el directorio `$LFS/sources`.
h. Modifique los permisos de acceso a este directorio usando el siguiente comando:

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

Descarga de componentes para la *toolchain*

i. Acceda al directorio `$LFS/sources` y descargue a él los archivos `wget-list` y `md5sums` desde `http://www.linuxfromscratch.org/lfs/downloads/8.1/` usando `wget`.
j. Descargue todos los paquetes que se necesitarán para construir el Linux mínimo usando el siguiente comando (en una única línea):

```
sudo wget --no-check-certificate --input-file=wget-list --continue
--directory-prefix=$LFS/sources
```

k. Revise que todo lo necesario se haya descargado usando los siguientes comandos:

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

l. Si todo está bien (recomendado un *snapshot*), proceda a crear el directorio `tools` adentro de `$LFS`.

Creación de y acceso al ambiente aislado

m. Cree un *link* simbólico hacia `$LFS/tools` en el directorio raíz usando el siguiente comando:

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

n. Cree un grupo de usuarios llamado `lfs` con el siguiente comando:

```
sudo groupadd lfs
```

o. Agregue un usuario llamado `lfs` a este grupo usando el siguiente comando:

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

- ¿Qué es `/dev/null`?
- Explique las opciones y parámetros usados en este comando.

p. Provea una contraseña fácil de recordar para este nuevo usuario con el siguiente comando:

```
sudo passwd lfs
```

q. Haga al usuario `lfs` el dueño de los directorios `$LFS/tools` y `$LFS/sources` con los siguientes comandos:

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

- ¿Qué significa ser dueño de un directorio?

r. *Logéese* con el usuario recién creado usando el siguiente comando:

```
su - lfs
```

- ¿Cuál es la diferencia entre una sesión de *login* y una de no-*login* (*login session vs. non-login session*)?

s. Cree un archivo que permita ejecutar una *shell* por defecto para el usuario `lfs`. Esta *shell* proveerá un nuevo ambiente en el que sólo estén definidas tres variables: el directorio HOME, el tipo de terminal que se usa y lo que se escribe en el *prompt* de la terminal (lo que sale siempre antes del cursor). Para hacerlo ejecute el siguiente comando (con cambios de línea):

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

- Explique exactamente qué se está haciendo con env –i HOME=$HOME TERM=$TERM PS1='\u:\w\$ ' /bin/bash.
- ¿Cuál es el efecto o propósito de preceder el comando anterior con exec?

t. Cree un último archivo que será leído para configurar el *shell* del inciso anterior cada vez que sea ejecutado. Use el siguiente comando (con cambios de línea):

```
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
```

Seguido de:

```
source ~/.bash_profile
```

Nota: en este punto su *prompt* debería aparecer verde. Puede obtener el efecto de este último comando saliendo e iniciando la sesión otra vez.

- `set +h` apaga la funcionalidad de *hashing* para la *shell*. ¿Para qué sirve el *hashing* de la *shell*?
- ¿Qué hace el comando `umask` y cuál es el efecto de usarlo con el parámetro `022`?

Instalación de *toolchain* para creación de un LFS

La *toolchain* que se creará requiere la instalación de las herramientas que se descargaron previamente. Todas estas herramientas vienen comprimidas en diferentes formatos, por lo que requerirá el programa `tar` para descomprimirlas. A continuación, se muestran las opciones que se necesitan para extraer archivos de cada formato de compresión:

```
tar -xvzf archivo.tar.gz

tar -xvjf archivo.tar.bz2

tar -xvJf archivo.tar.xz
```

El procedimiento general consistirá en extraer el paquete, ingresar al directorio donde se extrajo, crear el *script* de configuración del paquete, compilarlo, instalarlo, salir del directorio y borrar lo que se haya extraído y creado durante este proceso. **Importante:** las instrucciones dadas en este documento suponen, para cada paquete, que ya se ha extraído el paquete y se ha ingresado al directorio donde fue extraído. El orden de instalación es importante para la mayoría de las herramientas, y algunas requieren comandos de configuración relativamente extensos, por lo que se recomienda en estos casos crear un *script* de configuración usando algún editor de texto como `nano` y que, habiéndolo revisado y corregido, se ejecute usando el programa `bash`.

No olvide realizar *snapshots* de su progreso, por si alguna instalación sale mal. Se recomienda sacar *snapshots* al menos antes de iniciar cada instalación y luego de la segunda instalación de GCC; aunque mientras más frecuentes los *snapshots* mejor (pero mucho más espacio ocupado por la VM).

A. Si cerró la sesión o apagó su máquina virtual desde la creación del ambiente aislado, inicie sesión con el usuario `lfs` usando el siguiente comando:

```
su - lfs
```

Además, asegúrese de que la variable de ambiente `LFS` esté correctamente configurada en su ambiente, usando el siguiente comando (debería producir "`/mnt/lfs`"):

```
echo $LFS
```

Primera instalación de Binutils

B. Primera instalación de Binutils-2.29:
   a. Cree un directorio llamado `build` en `$LFS/sources/binutils-2.29` y acceda a él.
   b. Pese a la recomendación al principio de esta sección, haga una excepción con este único comando y escríbalo y ejecútelo en la terminal:

```
time { ../configure --prefix=/tools \
--with-sysroot=$LFS \
--with-lib-path=/tools/lib \
--target=$LFS_TGT \
--disable-nls \
--disable-werror \
&& make && make install; }
```

Nota: no es necesario incluir las diagonales invertidas. En *shell* sirven para introducir cambios de línea en un comando, y aquí se usan para aclarar que se trata de un mismo comando. Puede escribir todo de corrido.

   c. Anote los tiempos que se muestran al concluir el proceso. Este tiempo equivaldrá a 1 SBU, abreviatura para *standard build unit*. El tiempo que toma el resto de las instalaciones se mide aproximadamente con estas unidades.

- ¿Qué es la *target triplet* y cuál es su campo *vendor*?
- ¿Cuál es la importancia de este campo en la *target triplet* y para qué sirve que, en nuestro sistema, tenga el valor `lfs`?
- ¿Para qué sirve la opción `--prefix=/tools`?

Primera instalación de GCC

C. Primera instalación de GCC-7.2.0 (8.9 SBUs):
   a. Extraiga los paquetes `mpfr-3.1.5.tar.xz`, `gmp-6.1.2.tar.xz` y `mpc-1.0.3.tar.gz`; y coloque los directorios donde los extrajo dentro del directorio a

donde extrajo `gcc-7.2.0`. Cambie el nombre de cada directorio a `mpfr`, `gmp` y `mpc` respectivamente.

- ¿Qué hacen estos paquetes?

b. Ejecute el siguiente comando:

```
for file in gcc/config/{linux,i386/linux{,64}}.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
```

- ¿Qué hace el programa `sed`? Aclare con el mayor detalle posible qué está haciendo `sed` con ese relajo abajo de `cp -uv $file{,.orig}`.
- ¿Qué hace el comando `touch`?
- ¿Qué hace la opción `-u` de `cp` y cómo se relaciona con `touch` en este ciclo?
- ¿Para qué sirven las macros `STANDARD_STARTFILE_PREFIX_N` y por qué se les asignan los valores que se les asignan en este ciclo?
- Desglose y explique el funcionamiento de este ciclo.

c. Cree un directorio llamado `build` en `$LFS/sources/gcc-7.2.0` y acceda a él.

d. Ejecute el siguiente comando:

```
../configure \
--target=$LFS_TGT \
--prefix=/tools \
--with-glibc-version=2.11 \
--with-sysroot=$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-libmpx \
--disable-libquadmath \
--disable-libssp \
--disable-libvtv \
--disable-libstdcxx \
--enable-languages=c,c++
```

e. Compile e instale ejecutando primero `make` y luego `make install`.

- ¿Qué es un *sysroot* y por qué se detalla aquí con valor `$LFS`?
- ¿Por qué se especifica `$LFS_TGT` como el valor de `--target`?
- ¿Qué hace la opción `--disable-nls`?
- ¿Qué hace la opción `--disable-shared` y por qué es necesaria? ¿Cuál es la diferencia entre *linking* estático y dinámico?

D. Instalación de API de Linux (en `linux-4.12.7.tar.xz`, 0.1 SBUs):

a. Ejecute el siguiente comando:

```
make mrproper
```

b. Extraiga e instale los *headers* del paquete con los siguientes comandos:

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

Instalación de Glibc

E. Instalación de Glibc-2.26 (4.2 SBUs):

a. Cree un directorio en `$LFS/sources/glibc-2.26` llamado `build` y diríjase a él.

b. Ejecute el siguiente comando:

```
../configure \
--prefix=/tools \
--host=$LFS_TGT \
--build=$(../scripts/config.guess) \
--enable-kernel=3.2 \
--with-headers=/tools/include \
libc_cv_forced_unwind=yes \
libc_cv_c_cleanup=yes
```

- ¿Qué hace la ejecución del *script* `config.guess` en `glibc-2.21/scripts`?
- ¿Cuál es el efecto de las opciones `--host` y `--build` en este comando?
- ¿Qué es *stack unwinding* y qué es *forced stack unwinding*?

c. Compile e instale ejecutando `make` y luego `make install`.

Sanity check

d. Cree un programa en C llamado `dummy.c` cuyo `main()` no reciba parámetros y esté vacío.
e. Compile su programa usando, en lugar de sólo el comando `gcc`, `$LFS_TGT-gcc`.
f. Ejecute el siguiente comando:

```
readelf -l <archivo.o> | grep ': /tools'
```

donde `<archivo.o>` es el código objeto generado al compilar. Si no especificó un nombre con la opción `-o`, este archivo se llamará `a.out`.

g. Todo está bien si el resultado es lo siguiente:

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

Si todo está bien proceda a eliminar su programa en C y el archivo objeto generado. De lo contrario deberá revisar las instalaciones realizadas hasta ahora (posiblemente sea necesario volver a empezar) para corregir el problema.

- ¿Qué es un archivo ELF y qué hace la opción `-l` de `readelf`?
- Hasta el momento instalamos GCC y Binutils con configuraciones que conforman lo que el libro llama *cross GCC* y *cross Binutils*. ¿Qué es un *cross compiler*? ¿Por qué instalamos versiones *cross* de GCC y Binutils? Se recomienda leer la sección 5.2 del libro de LFS.
- ¿Cuál es el efecto de compilar con `$LFS_TGT-gcc` en lugar de sólo `gcc`?

F. Instalación de Libstdc++-7.2.0 (0.4 SBUs):

a. Vuelva a extraer `gcc-7.2.0` y acceda al directorio donde fue extraído.
b. Cree un directorio llamado `build` y diríjase a él.
c. Ejecute el siguiente comando:

```
../libstdc++-v3/configure \
--host=$LFS_TGT \
--prefix=/tools \
--disable-multilib \
--disable-nls \
--disable-libstdcxx-threads \
--disable-libstdcxx-pch \
--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/7.2.0
```

d. Compile e instale usando `make` y luego `make install`.

- Explique la relación entre Binutils, GCC, Glibc y Libstdc++.
- ¿Por qué se usa la opción `--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/7.2.0`?

Segunda instalación de Binutils

G. Segunda instalación de Binutils-2.29 (1.1 SBU):
   a. Cree un directorio en `$LFS/sources/binutils-2.29` llamado `build` y diríjase a él.
   b. Ejecute el siguiente comando:

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

      - ¿Por qué se debe realizar una segunda instalación de Binutils?
   c. Compile e instale usando `make` y luego `make install`.
   d. Ejecute los siguientes comandos:

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

Segunda instalación de GCC

H. Segunda instalación de GCC-7.2.0 (11 SBUs)
   a. Ejecute el siguiente comando:

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

      - ¿Para qué sirve el comando `dirname`?
      - ¿Cuál es la diferencia entre ejecutar algo con `$` y ejecutarlo con `` ` ``?
      - ¿Qué contiene el *header* `limits.h`? Normalmente GCC ejecuta este mismo comando durante su instalación, y el resultado es un *header* llamado `limits.h` que incluye otro *header* llamado `limits.h` pero provisto por el sistema. ¿Por qué la primera instalación de GCC no hizo esto desde un principio?
   b. Vuelva a ejecutar el siguiente comando:

```
for file in gcc/config/{linux,i386/linux{,64}}.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
```

c. Vuelva a extraer `mpfr-3.1.5.tar.xz`, `gmp-6.1.2.tar.xz`, `mpc-1.0.3.tar.gz`, y cambie el nombre de los directorios de extracción a `mpfr`, `gmp` y `mpc` respectivamente.
d. Vuelva a crear un directorio llamado `build` en `$LFS/sources/gcc-7.2.0`, y diríjase a él.
e. Ejecute el siguiente comando:

```
CC=$LFS_TGT-gcc \
CXX=$LFS_TGT-g++ \
AR=$LFS_TGT-ar \
RANLIB=$LFS_TGT-ranlib \
../configure \
--prefix=/tools \
--with-local-prefix=/tools \
--with-native-system-header-dir=/tools/include \
--enable-languages=c,c++ \
--disable-libstdcxx-pch \
--disable-multilib \
--disable-bootstrap \
--disable-libgomp
```

f. Compile e instale usando `make` y luego `make install`.
g. Cree un *link* simbólico hacia `gcc` llamado `/tools/bin/cc` (el *link* sólo se llama `cc`, pero debe estar ubicado en `/tools/bin`).

   - `--disable-bootstrap` deshabilita el comportamiento habitual que tiene GCC de compilarse haciendo un *bootstrap build*. ¿Qué es esto?

h. Repita el procedimiento de verificación de compilación donde se crea y compila un programa en C con el `main` vacío y luego se ejecuta `readelf` sobre el código objeto producido, pero en lugar de compilar con `$LFS_TGT-gcc` compile con `cc` (ver inciso E.). Los resultados deberían ser los mismos. Si no obtiene el resultado esperado o no obtiene ningún resultado, compruebe que creó el *link* simbólico del inciso anterior y verifique que `/tools/bin` aparece al principio del *output* de echo `$PATH`.

I. Instalación de Tcl-core-8.6.7 (0.4 SBUs):

a. Acceda al directorio `unix` y ejecute el siguiente comando:

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

b. Compile e instale usando `make` y luego `make install`.

- ¿Qué es Tcl y con qué propósito fue creado?

c. Agregue permisos de escritura para el *owner* sobre el archivo `/tools/lib/libtcl8.6.so` usando el siguiente comando:

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

d. Ejecute el siguiente comando:

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

e. Cree un *link* simbólico en hacia `tclsh8.6` llamado `/tools/bin/tclsh`.

J. Instalación de Expect-5.45 (0.1 SBUs):

a. Cree una copia de `configure` llamada `configure.orig`.
b. Usando `sed` substituya las ocurrencias de `/usr/local/bin` por `/bin` en `configure`.
c. Ejecute el siguiente comando:

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

d. Compile usando `make` e instale usando `make SCRIPTS="" install`.

- ¿Para qué sirve Expect?

A partir de este punto, casi todos los paquetes se instalarán siguiendo el siguiente patrón:

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

make

make install
```

Si se requieren cambios a este procedimiento o las instrucciones de instalación son diferentes, se especificará en el inciso correspondiente. Tome en cuenta que este patrón de instalación se lleva a cabo luego de extraer el paquete y acceder a su directorio de extracción. Tampoco olvide borrar el directorio de extracción y cualquier otro directorio creado, luego de la instalación.

K. Instale DejaGNU-1.6 (0.1 SBUs).

- ¿Cuál es la relación entre DejaGNU, Expect y Tcl?

L. Instale Check-0.11.0 (0.1 SBUs) reemplazando `./configure --prefix=/tools` con

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

Note el espacio entre '=' y '.'.

- ¿Qué es `pkg-config` y por qué se especifica una variable `PKG_CONFIG` vacía en esta configuración?

M. Ejecute el siguiente comando:

```
sed -i s/mawk// configure
```

Luego instale Ncurses-6.0 (0.5 SBUs) agregando, además de `--prefix=/tools`, las siguientes opciones:

```
--with-shared \
--without-debug \
--without-ada \
--enable-widec \
--enable-overwrite
```

- ¿Para qué sirve Ncurses?

N. Instalación de Bash-4.4 (0.4 SBUs):
   a. Instale Bash-4.4 añadiendo, además de `--prefix=/tools`, la opción `--without-bash-malloc`.
      - ¿Qué es `malloc`?
   b. Cree un *link* simbólico hacia `bash` llamado `/tools/bin/sh`.

O. Instale M4-1.4.18 (0.2 SBUs).

- ¿Para qué sirve M4?

P. Instale Bison-3.0.4 (0.3 SBUs).
Q. Instalación de Bzip2-1.0.6 (menos de 0.1 SBUs):
   a. Ejecute `make`.
   b. Ejecute `make PREFIX=/tools install`.
R. Instale Coreutils-8.27 (0.6 SBUs) añadiendo, además de `--prefix=/tools`, la opción `--enable-install-program=hostname`.

- ¿Qué hace el programa `hostname` de Coreutils?

S. Instale Diffutils-3.6 (0.2 SBUs).
T. Instale File-5.31 (0.1 SBUs).
U. Instale Findutils-4.6.0 (0.3 SBUs).
V. Instale Gawk-4.1.4 (0.2 SBUs).
W. Instalación de Gettext-0.19.8.1 (0.8 SBUs):

a. Acceda al directorio `gettext-tools`.
b. Ejecute el siguiente comando:

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

c. Compile los programas `msgfmt`, `msgmerge` y `xgettext` con los siguientes comandos:

```
make -C gnulib-lib
make -C intl pluralx.c
make -C src msgfmt
make -C src msgmerge
make -C src xgettext
```

d. Copie los archivos `msgfmt`, `msgmerge` y `xgettext` del directorio `src` a `/tools/bin`.

- ¿Cuál es la relación entre los programas `msgfmt`, `msgmerge` y `xgettext` de Gettext?

X. Instale Grep-3.1 (0.2 SBUs).
Y. Instale Gzip-1.8 (0.1 SBUs).
Z. Instale Make-4.2.1 (0.1 SBUs) añadiendo, además de `--prefix=/tools`, la opción `--without-guile`.

- ¿Qué es GNU Guile?

AA. Instale Patch-2.7.5 (0.2 SBUs).

- ¿Cuál es la relación entre Patch y Diffutils?

BB. Instalación de Perl-5.26.0 (1.3 SBUs):

a. Ejecute el siguiente comando:

```
sed -e '9751 a#ifndef PERL_IN_XSUB_RE' \
-e '9808 a#endif' \
-i regexec.c
```

b. Ejecute el siguiente comando:

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

c. Compile con `make`.
d. Ejecute el siguiente comando:

```
cp -v perl cpan/podlators/scripts/pod2man /tools/bin
```

e. Cree los directorios `/tools/lib/perl5` y `/tools/lib/perl5/5.26.0`.

f. Copie recursivamente todo lo que esté en el directorio `lib`, dentro del directorio donde extrajo Perl, a `/tools/lib/perl5/5.26.0`.

CC. Instale Sed-4.4 (0.2 SBUs).

DD. Instale Tar-1.29 (0.3 SBUs).

EE. Instale Texinfo-6.4 (0.2 SBUs) (ignore un error que le aparecerá referente a `TestXS_la-TestXS.lo`).

FF. Instale Util-linux-2.30.1 (0.8 SBUs) añadiendo, además de `--prefix=/tools`, las siguientes opciones:

```
--without-python \
--disable-makeinstall-chown \
--without-systemdsystemunitdir \
--without-ncurses \
PKG_CONFIG=""
```

- ¿Qué es la filosofía UNIX?
- ¿Por qué hay quienes consideran que `systemd` va en contra de esta filosofía?

GG. Instale Xz-5.2.3 (0.2 SBUs).

HH. Ejecute los siguientes comandos:

```
strip --strip-debug /tools/lib/*
/usr/bin/strip --strip-unneeded /tools/{,s}bin/*
rm -rf /tools/{,share}/{info,man,doc}
```

- ¿Qué son *debugging symbols*?
- ¿Para qué sirve `strip`?
- ¿Por qué el primer comando simplemente ejecuta `strip`, pero el segundo ejecuta `/usr/bin/strip`?
- Explique el concepto de *relocation* asociado a los *linkers*, y su relación con la opción `--strip-unneeded`?

II. Salga de la sesión que inició con el usuario `lfs` mediante el comando `exit`.

JJ. Cambie recursivamente la pertenencia de `$LFS/tools` al usuario `root` y grupo `root` usando el siguiente comando:

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

Construcción de LFS (parte I)

a. Comience por crear los directorios `$LFS/dev`, `$LFS/proc`, `$LFS/sys` y `$LFS/run`.

b. A continuación, ejecute los siguientes comandos:

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

c. Monte el sistema de archivos `/dev` usando el siguiente comando:

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

- ¿Qué significa "montar" un sistema de archivos?

d. Ahora monte los sistemas de archivos virtuales restantes:

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

- ¿Qué es un sistema de archivos? ¿Qué es un sistema de archivos virtual?
- Explique el propósito de cada uno de los sistemas de archivos `/dev`, `/proc`, `/sysfs` y `/run`.
- Explique a detalle qué hacen los comandos del inciso *b*.
- Explique a detalle qué hacen los comandos `mount` de este inciso.

e. Ejecute el siguiente comando para verificar la existencia del *link* simbólico `$LFS/dev/shm` y crear el directorio al que apunta en caso de que sí exista:

```
if [ -h $LFS/dev/shm ]; then
     sudo mkdir -pv $LFS/$(readlink $LFS/dev/shm)
fi
```

f. Ahora ejecute el siguiente comando:

```
sudo 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
```

- Explique exactamente qué hace el comando anterior y por qué es importante para el resto del proyecto.

Sanity check

En este punto es importante realizar un paso de revisión que el libro de LFS no incluye, pero cuyo potencial problema nos impedirá continuar más adelante. Cree un programa de prueba en C con el siguiente comando:

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

Ahora compílelo usando el comando `cc`:

```
cc dummy.c
```

Esto no debería producir ningún error. A continuación, diríjase al directorio `/tools/i686-pc-linux-gnu/bin` y ejecute el siguiente comando:

```
readelf -e as | grep interpreter
```

El resultado debería ser el siguiente:

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

Si el resultado no incluye `/tools`, el ensamblador instalado previamente no fue correctamente configurado. Lamentablemente, este es un paso crucial realizado temprano en la creación del *toolchain*, por lo que será necesario repetir toda la construcción de la *toolchain* (capítulo 5 del libro de LFS) antes de continuar (específicamente, hay que repetir la primera instalación de Binutils ya que este paquete incluye el ensamblador, pero esa instalación es la primera que hacemos en la creación del *toolchain*, y todo lo que instalamos después depende de ella).

Si todo está bien proceda a eliminar `dummy.c` y `a.out` del directorio donde los creó. **Nota importante**: si salimos del ambiente creado en el inciso más reciente, será necesario volver a entrar a él antes de continuar con cualquier paso a partir de este punto. Esto significa que habrá que volver a ejecutar el comando `chroot` de este documento. Si además de salir del ambiente reiniciamos la máquina virtual (o de alguna otra forma desmontamos los sistemas de archivos virtuales), deberemos volver a montar los sistemas de archivos virtuales como en los incisos *c* y *d* antes de ejecutar el `chroot`.

g. Use los siguientes comandos para crear una variedad de directorios y *links* simbólicos necesarios:

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

case $(uname -m) in
  x86_64) mkdir -v /lib64 ;;
esac

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

```
ln -sv /tools/bin/{bash,cat,dd,echo,ln,pwd,rm,stty} /bin
ln -sv /tools/bin/{install,perl} /usr/bin
```

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

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

- ¿Por qué es necesario crear ese segundo conjunto de *links* simbólicos que apuntan a elementos en `/tools/bin`?
- ¿Para qué sirve el comando `install`?
- ¿Para qué sirve el archivo `/etc/mtab`, y por qué se *linkea* a `/proc/self/mounts`?
- ¿Qué es el *Filesystem Hierarchy Standard* (FHS)?

h. Cree y llene el archivo `/etc/passwd` con el siguiente comando:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/bin/false
daemon:x:6:6:Daemon User:/dev/null:/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/var/run/dbus:/bin/false
nobody:x:99:99:Unprivileged User:/dev/null:/bin/false
EOF
```

i. Cree y llene el archivo `/etc/group` con el siguiente comando:

```
cat > /etc/group << "EOF"
root:x:0:
bin:x:1:daemon
sys:x:2:
kmem:x:3:
tape:x:4:
tty:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
adm:x:16:
messagebus:x:18:
systemd-journal:x:23:
input:x:24:
mail:x:34:
nogroup:x:99:
```

```
users:x:999:
EOF
```

j. Ejecute el siguiente comando:

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

k. Cree los siguientes archivos y modifique sus permisos con los siguientes comandos:

```
touch /var/log/{btmp,lastlog,faillog,wtmp}
chgrp -v utmp /var/log/lastlog
chmod -v 664 /var/log/lastlog
chmod -v 600 /var/log/btmp
```

- ¿Para qué sirven los archivos `/etc/passwd` y `/etc/group`? Explique el formato de estos archivos.
- Explique el propósito y configuración de los usuarios creados en `/etc/passwd`.
- ¿Cuál es el efecto visible del comando ejecutado en este inciso, y por qué se ejecuta usando `exec` en lugar de sólo `/tools/bin/bash`?

Instalación de herramientas en ambiente aislado

Del siguiente inciso en adelante instalaremos varias herramientas en nuestro ambiente aislado. Los paquetes que contienen estas herramientas están en `/sources`. Asegúrese de seguir el procedimiento de instalación recomendado (descomprimir paquete, acceder a directorio, seguir instrucciones, salir y borrar).

l. Instalación de Linux-4.12.7 API *Headers* (ubicados en `linux-4.12.7.tar.xz`, 0.1 SBUs):
   1. Limpie el ambiente de instalación con el siguiente comando:

```
make mrproper
```

   2. Compile los *headers* de Linux y cópielos a su ubicación correspondiente (excluyendo ciertos archivos innecesarios) con los siguientes comandos:

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

m. Instalación de Man-pages-4.12 (0.1 SBUs):
   1. Ejecute `make install`.

Instalación de Glibc

n. Instalación de Glibc-2.26 (20 SBUs):
   1. Comience por aplicar un parche al paquete con el siguiente comando:

```
patch -Np1 -i ../glibc-2.26-fhs-1.patch
```

- Dé un vistazo al contenido del archivo `glibc-2.26-fhs-1.patch`, y responda: ¿qué hace el programa `patch`?
- ¿Para qué sirven las opciones `-Np1` e `-i` utilizadas?

2. Cree un *link* simbólico a `/tools/lib/gcc` en `/usr/lib` con el siguiente comando:

```
ln -sfv /tools/lib/gcc /usr/lib
```

3. Ejecute el siguiente comando para creación de algunos directorios y *links* simbólicos necesarios:

```
case $(uname -m) in
  i?86)        GCC_INCDIR=/usr/lib/gcc/$(uname -m)-pc-linux-gnu/7.2.0/include
               ln -sfv ld-linux.so.2 /lib/ld-lsb.so.3
  ;;
  x86_64)      GCC_INCDIR=/usr/lib/gcc/x86_64-pc-linux-gnu/7.2.0/include
               ln -sfv ../lib/ld-linux-x86-64.so.2 /lib64
               ln -sfv ../lib/ld-linux-x86-64.so.2 /lib64/ld-lsb-x86-64.so.3
  ;;
esac
```

4. Elimine un archivo que pudo haber quedado de intentos de compilación previos:

```
rm -fv /usr/include/limits.h
```

5. Cree y acceda a un directorio llamado `build`.
6. Prepare la compilación de Glibc con el siguiente comando:

```
CC="gcc -isystem $GCC_INCDIR -isystem /usr/include" \
../configure \
      --prefix=/usr \
      --disable-werror \
      --enable-kernel=3.2 \
      --enable-stack-protector=strong \
      libc_cv_slibdir=/lib
```

7. Elimine la variable de entorno GCC_INCDIR mediante el siguiente comando:

```
unset GCC_INCDIR
```

8. Compile usando `make`.
9. Ejecute la *test suite* de Glibc usando `make check`. Obtendrá algunos *tests* fallidos, pero mientras no se trate de un número muy grande (más de diez) no habrá problema.
10. Ejecute el siguiente comando:

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

- ¿Qué hace `touch` cuando recibe un archivo inexistente como argumento?

11. Ejecute el siguiente comando para corregir un problema con el *Makefile* generado:

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

12. Instale usando `make install`.
13. Instale el archivo de configuración y el directorio de ejecución de `nscd` con los siguientes comandos:

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

    - ¿Qué es `nscd`?

14. Este inciso instalará los `locales` necesarios para que los *tests* ejecutados más adelante pasen las pruebas necesarias. Tome en cuenta que hay dos caminos: el primero consiste en ejecutar el siguiente comando:

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

    Esto instala varios `locales` adicionales a los mínimos necesarios y es "tardado" (en realidad no se tarda tanto). El otro camino consiste en la instalación manual de los `locales` mínimos necesarios con los siguientes comandos:

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

    - Independientemente del camino que haya tomado responda: ¿qué es un `locale`?

15. Cree y llene el archivo `/etc/nsswitch.conf` con el siguiente comando:

```
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
```

    - ¿Cuál es el propósito de `nsswitch.conf`?
    - ¿Cuál es el propósito de cada una de las bases de datos configuradas en este archivo en el comando anterior?
16. Ejecute los siguientes comandos para instalar los datos de zonas temporales:

```
tar -xf ../../tzdata2017b.tar.gz

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

for tz in etcetera southamerica northamerica europe \
      africa antarctica asia australasia backward \
      pacificnew systemv; do

      zic -L /dev/null -d $ZONEINFO -y "sh yearistype.sh" ${tz}
      zic -L /dev/null -d $ZONEINFO/posix -y "sh yearistype.sh" ${tz}
      zic -L leapseconds -d $ZONEINFO/right -y "sh yearistype.sh" ${tz}

done

cp -v zone.tab zone1970.tab iso3166.tab $ZONEINFO
zic -d $ZONEINFO -p America/New_York
unset ZONEINFO
```

17. Obtenga la zona temporal adecuada para su sistema ejecutando y siguiendo los pasos del comando `tzselect`.
    - ¿Para qué sirve el comando `zic`?
    - ¿Cuál es el propósito de los archivos `zone.tab`, `zone1970.tab` e `iso3166.tab`?
18. Copie la zona temporal dada por el inciso anterior con el siguiente comando:

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

    donde `<xxx>` es la zona temporal dada por el inciso anterior.

19. Ejecute el siguiente comando para agregar directorios en los que el *dynamic loader* pueda encontrar bibliotecas donde varios programas las instalan por defecto, incluyendo directorios especificados en archivos de configuración:

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

# Add an include directory
include /etc/ld.so.conf.d/*.conf
EOF
mkdir -pv /etc/ld.so.conf.d
```

o. Ejecute los siguientes comandos para hacer un *backup* del *linker* instalado en `/tools`, reemplazarlo por el *linker* creado durante la segunda instalación de Binutils-2.29 en la construcción de la *toolchain*, y crear un *link* simbólico a dicho *linker*:

```
mv -v /tools/bin/{ld,ld-old}
mv -v /tools/$(uname -m)-pc-linux-gnu/bin/{ld,ld-old}
mv -v /tools/bin/{ld-new,ld}
ln -sv /tools/bin/ld /tools/$(uname -m)-pc-linux-gnu/bin/ld
```

p. Modifique las especificaciones de GCC para que se adapten al uso de este nuevo *linker,* con el siguiente comando:

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

- ¿Qué hace la instrucción `n` en `sed`?
- ¿Cuál es el propósito del archivo `specs`?

Sanity check

q. Ahora vamos a ejecutar varios comandos para verificar que todo esté como debe estar. Primero cree otro programa de prueba con el siguiente comando:

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

Y luego compílelo con el siguiente comando:

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

Si no hay errores ejecute lo siguiente:

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

Y el resultado debería ser el siguiente:

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

- ¿Qué hace la opción `-Wl` en el comando `cc`?
- ¿Para qué sirve usar `&>`?

r. A continuación, buscaremos varios patrones en `dummy.log`. Primero ejecute lo siguiente:

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

El resultado debería ser el siguiente:

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

Luego ejecute:

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

Y el resultado debería ser:

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

A continuación, realizaremos tres intercambios más de ejecución versus resultado esperado:

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

```
...
SEARCH_DIR("/usr/lib")
SEARCH_DIR("/lib");
...
```

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

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

```
grep found dummy.log
```

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

Si todos los resultados esperados han sido satisfechos proceda a eliminar los archivos creados `dummy.c`, `a.out` y `dummy.log`.

- ¿Cuál es el efecto de los símbolos `.`, `[]` y `*` en los argumentos de `grep`?
- ¿Qué hace la opción `-o` de `grep`?

- Explique el propósito de los archivos `crt1.o`, `crti.o` y `crtn.o`
- En el comando `grep` con la opción `-B1`, ¿qué sucedería si no usamos esa opción?
- De acuerdo a estos resultados, ¿dónde se inicia la búsqueda de encabezados que #incluimos en nuestros programas?
- ¿Qué es, y para qué sirve un *soname*? Explique qué son `libc.so.6` y `ld-linux.so.2` (se recomienda investigar el concepto de *shared library*).

s. Instalación de Zlib-1.2.11 (0.1 SBUs):
   1. Para cada paquete a partir de éste, por separado y en su directorio correspondiente, ejecute el siguiente comando:

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

   2. Luego compile usando `make`.
   3. A continuación, ejecute la *test suite* con `make check`.
   4. Instale el paquete con `make install`.
   5. Ejecute los siguientes comandos:

```
mv -v /usr/lib/libz.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libz.so) /usr/lib/libz.so
```

t. Instale File-5.31 (0.1 SBUs) siguiendo los pasos 1-4 del inciso anterior. **Estos serán los pasos de instalación por defecto, de ahora en adelante**.

u. Instalación de Readline-7.0 (0.1 SBUs):
   1. Ejecute los siguientes comandos para evitar un posible error luego de la instalación:

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

   2. A continuación, cree el *Makefile* con el siguiente comando:

```
./configure --prefix=/usr \
--disable-static \
--docdir=/usr/share/doc/readline-7.0
```

   3. Compile usando `make SHLIB_LIBS="-L/tools/lib -lncursesw"`. Luego instale con `make SHLIB_LIBS="-L/tools/lib -lncursesw" install`.
      - ¿Qué es *underlinking*?

4. Las siguientes instrucciones cambian la ubicación de las *dynamic libraries* instaladas, e instalan la documentación:

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

ln -sfv ../../lib/$(readlink /usr/lib/libreadline.so) \
/usr/lib/libreadline.so

ln -sfv ../../lib/$(readlink /usr/lib/libhistory.so) \
/usr/lib/libhistory.so

install -v -m644 doc/*.{ps,pdf,html,dvi} \
/usr/share/doc/readline-7.0
```

v. Instale M4-1.4.18 (0.4 SBUs).
w. Instalación de Bc-1.07.1 (0.1 SBUs):

1. Los siguientes comandos corrigen un archivo interno:

```
cat > bc/fix-libmath_h << "EOF"
#! /bin/bash
sed -e '1 s/^/{"/' \
-e 's/$/",/' \
-e '2,$ s/^/"/' \
-e '$ d' \
-i libmath.h

sed -e '$ s/$/0}/' \
-i libmath.h

EOF
```

2. Los siguientes comandos crean unos *links* simbólicos necesarios para la instalación actual y para el correcto funcionamiento del paquete al concluir esta fase de instalaciones:

```
ln -sv /tools/lib/libncursesw.so.6 /usr/lib/libncursesw.so.6
ln -sfv libncurses.so.6 /usr/lib/libncurses.so
```

3. Ejecute el siguiente comando para corregir el *script* `configure`:

```
sed -i -e '/flex/s/as_fn_error/: ;; # &/' configure
```

4. Configure el paquete con la siguiente instrucción:

```
./configure --prefix=/usr \
--with-readline \
--mandir=/usr/share/man \
--infodir=/usr/share/info
```

5. Compile el paquete con `make`.
6. El chequeo de este paquete no se realiza con `make check` sino con la siguiente instrucción:

```
echo "quit" | ./bc/bc -l Test/checklib.b > resultados
```

7. Como resultado deberá obtener un archivo llamado `resultados`. Revíselo usando `cat resultados | grep failed` y confirme que ningún porcentaje de fallos sea mayor a 1 (la mayoría deberían ser cero).
8. Instale con `make install`.

Instalación de Binutils

x. Instalación de Binutils-2.29 (5.8 SBUs):

1. Ejecute el siguiente comando:

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

Debería obtener el siguiente resultado:

```
spawn ls
```

Si obtiene el texto:

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

hay un problema, y deberá regresar y repetir los pasos que sean necesarios para evitarlo.

- Investigue:
  - ¿En qué consiste una PTY?
  - ¿Cómo funcionan las PTY?
  - ¿Cómo se relacionan con `expect`?
  - ¿Cómo se relacionan con el sistema de archivos `devpts` que montamos casi al inicio de la construcción del LFS?
  - ¿Cómo se relacionan con los emuladores de terminales?

2. Cree un directorio llamado `build` y acceda a él.
3. Ejecute el siguiente comando de configuración:

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

4. Compile usando el siguiente comando:

```
make tooldir=/usr
```

5. Ejecute la *test suite* con el siguiente comando (es probable que falle un *test* con el archivo `debug_msg.sh`, pero no es problema):

```
make -k check
```

   - ¿Qué hace la opción `-k`?

6. Instale usando el siguiente comando:

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

y. Instalación de GMP-6.1.2 (1.2 SBUs):

1. Configure el paquete para compilación con el siguiente comando:

```
./configure \
--prefix=/usr \
--enable-cxx \
--disable-static \
--docdir=/usr/share/doc/gmp-6.1.2
```

2. Compile el paquete y la documentación con los siguientes comandos:

```
make
make html
```

3. Ejecute la *test suite* con el siguiente comando:

```
make check 2>&1 | tee gmp-check-log
```

   Todos los *tests* (190) deben pasar. Para revisar este resultado ejecute:

```
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

4. Instale el paquete y la documentación con los siguientes comandos:

```
make install
make install-html
```

   - ¿Qué diferencia hay entre usar `&>` y `2>&1`, y cuál es el efecto de esto último combinado con el comando `tee`?

z. Instalación de MPFR-3.1.5 (0.8 SBUs):

1. Configure el paquete para compilación con el siguiente comando:

```
./configure --prefix=/usr \
--disable-static \
--enable-thread-safe \
```

```
--docdir=/usr/share/doc/mpfr-3.1.5
```

2. Compile el paquete y la documentación con `make` y `make html`.
3. Ejecute la *test suite* con `make check`.
4. Instale el paquete y la documentación con `make install` y `make install-html`.

aa. Instalación de MPC-1.0.3 (0.3 SBUs):

1. Configure el paquete con el siguiente comando:

```
./configure --prefix=/usr \
--disable-static \
--docdir=/usr/share/doc/mpc-1.0.3
```

2. Compile el paquete y la documentación con `make` y `make html`.
3. Ejecute la *test suite* con `make check`.
4. Instale el paquete y la documentación con `make install` y `make install-html`.

Instalación de GCC

bb. Instalación de GCC-7.2.0 (82 SBUs):

1. Comience por eliminar el *link* simbólico que se creó en el inciso *n*.

```
rm -vf /usr/lib/gcc
```

2. Cree un directorio llamado `build` y acceda a él.
3. Prepare el paquete para compilación con el siguiente comando:

```
SED=sed \
../configure \
--prefix=/usr \
--enable-languages=c,c++ \
--disable-multilib \
--disable-bootstrap \
--with-system-zlib
```

4. Compile usando `make` (toma un buen tiempo).
5. Un *test* agota la pila del sistema. Ajuste el tamaño del *stack* usando el siguiente comando:

```
ulimit -s 32768
```

6. Inicie la *test suite* con el siguiente comando (esto toma mucho tiempo, así que procure dejar la máquina virtual en ejecución durante toda la noche; y si no ha hecho *snapshots*

este es un buen momento para hacer uno):

```
make -k check
```

Los resultados de los *tests* pueden ser diferentes según la plataforma.

Puede revisar los resultados de las pruebas con el comando siguiente:

```
../contrib/test_summary | grep -A7 Summ
```

Es de esperarse que se reporten algunos pocos fallos no esperados en los resultados. Usualmente son menos de diez fallos inesperados por conjunto de pruebas, cuando ocurren.

7. Instale con `make install`.
8. Ejecute los siguientes comandos para crear *links* simbólicos requeridos por algunos paquetes:

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

9. Ejecute los siguientes comandos para crear un *link* simbólico (y el directorio que lo contendrá) que permitirá el uso de *Link Time Optimization* (LTO):

```
install -v -dm755 /usr/lib/bfd-plugins
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/7.2.0/liblto_plugin.so \
/usr/lib/bfd-plugins/
```

- ¿Qué es y para qué sirve LTO?

Sanity check

10. A continuación, se realizarán pruebas similares a las que se hicieron para probar los ajustes al *toolchain*. Comience por crear y compilar un programa de prueba:

```
echo 'int main(){}' > dummy.c
cc dummy.c -v -Wl,--verbose &> dummy.log
```

Si no hay errores ejecute:

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

Y el resultado debería ser el siguiente:

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

Luego ejecute:

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

El resultado debería ser el siguiente:

```
/usr/lib/gcc/i686-pc-linux-gnu/7.2.0/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/7.2.0/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/7.2.0/../../../crtn.o succeeded
```

Verifique los directorios de búsqueda para `#include` con el siguiente comando:

```
grep -B4 '^ /usr/include' dummy.log
```

Y el resultado debería ser:

```
#include <...> search starts here:
/usr/lib/gcc/i686-pc-linux-gnu/7.2.0/include
/usr/local/include
/usr/lib/gcc/i686-pc-linux-gnu/7.2.0/include-fixed
/usr/include
```

A continuación, realizaremos tres intercambios más de ejecución versus resultado esperado:

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

```
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib32")
SEARCH_DIR("/usr/local/lib32")
SEARCH_DIR("/lib32")
SEARCH_DIR("/usr/lib32")
SEARCH_DIR("/usr/i686-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

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

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

```
grep found dummy.log
```

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

Si todo sale como esperado, proceda a eliminar los archivos creados para las pruebas (`dummy.c`, `a.out` y `dummy.log`). Termine ejecutando los siguientes comandos para reubicar unos archivos:

```
mkdir -pv /usr/share/gdb/auto-load/usr/lib
mv -v /usr/lib/*gdb.py /usr/share/gdb/auto-load/usr/lib
```

cc. Instalación de Bzip2-1.0.6 (0.1 SBUs)

1. Aplique un parche necesario con el siguiente comando:

```
patch -Np1 -i ../bzip2-1.0.6-install_docs-1.patch
```

2. Ejecute el siguiente comando para que los *links* simbólicos instalados sean relativos:

```
sed -i 's@\(ln -s -f \)$(PREFIX)/bin/@\1@' Makefile
```

   - Explique el propósito y funcionamiento de `\1` en el comando anterior.

3. Modifique el directorio de instalación de las *man pages* con la siguiente instrucción:

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

4. Ejecute los siguientes comandos para preparar la compilación:

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

5. Compile con `make`.
6. Instale con `make PREFIX=/usr install`.
7. Realice algunas reubicaciones necesarias para finalizar la instalación:

```
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
```

dd. Instalación de Pkg-config-0.29.2 (0.3 SBUs):

1. Ejecute el siguiente comando de configuración:

```
./configure \
--prefix=/usr \
--with-internal-glib \
--disable-host-tool \
--docdir=/usr/share/doc/pkg-config-0.29.2
```

   - ¿Para qué sirve Pkg-config?

2. Compile con `make`.

3. Ejecute la *test suite* con `make check`.
4. Si todo está bien, instale con `make install`.

ee. Instalación de Ncurses-6.0 (0.3 SBUs):

1. Con la siguiente instrucción, omita la instalación de una biblioteca estática que no debe ser parte de la configuración:

```
sed -i '/LIBTOOL_INSTALL/d' c++/Makefile.in
```

2. Configure usando la siguiente instrucción:

```
./configure --prefix=/usr \
--mandir=/usr/share/man \
--with-shared \
--without-debug \
--without-normal \
--enable-pc-files \
--enable-widec
```

3. Compile usando `make`.
4. Instale usando `make install`.
5. Ejecute los siguientes comandos para reubicar las bibliotecas compartidas a donde deben estar y redirigir un *link* simbólico, correspondientemente:

```
mv -v /usr/lib/libncursesw.so.6* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libncursesw.so) \
/usr/lib/libncursesw.so
```

6. Ejecute las siguientes instrucciones para asegurar que aplicaciones que funcionan sin *wide-character libraries* (que se instalaron al usar `--enable-widec`) usen las *libraries* instaladas:

```
for lib in ncurses form panel menu ; do
    rm -vf /usr/lib/lib${lib}.so
    echo "INPUT(-l${lib}w)" > /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc /usr/lib/pkgconfig/${lib}.pc
done
```

   - Lea el contenido de uno de los archivos creados con la instrucción anterior (*e.g.* usando `cat /usr/lib/libncurses.so`). ¿Qué hace la instrucción allí contenida?
   - ¿Qué es un *linker script* y para qué sirve?

7. Con el similar propósito de obligar a que las aplicaciones que requieren Curses usen el paquete recién instalado, ejecute los siguientes comandos:

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

- ¿Cuál es la diferencia entre Curses y Ncurses?

8. Termine instalando la documentación del paquete:

```
mkdir -v /usr/share/doc/ncurses-6.0
cp -v -R doc/* /usr/share/doc/ncurses-6.0
```

ff. Instalación de Attr-2.4.47 (menos de 0.1 SBUs):

1. Ejecute la siguiente instrucción para que el directorio de documentación incluya la versión del paquete:

```
sed -i -e 's|/@pkg_name@|&-@pkg_version@|' include/builddefs.in
```

2. Anteriormente se instaló un paquete de *man pages*. Evite reinstalar documentación redundante con la siguiente instrucción:

```
sed -i -e "/SUBDIRS/s|man[25]||g" man/Makefile
```

3. Use el siguiente comando para corregir un problema en los procedimientos de *test*:

```
sed -i 's:{(:\\{(:' test/run
```

4. Configure para compilación, compile, ejecute los *tests*; e instale (si los *tests* son pasados) con las siguientes instrucciones:

```
./configure --prefix=/usr \
--bindir=/bin \
--disable-static

make
make -j1 tests root-tests
make install install-dev install-lib
chmod -v 755 /usr/lib/libattr.so
```

- Investigue qué son las *automatic variables* en un *makefile* y explique el funcionamiento de los últimos dos comandos `make`.

5. Mueva la biblioteca compartida instalada a `/lib` y cree un *link* simbólico en su lugar:

```
mv -v /usr/lib/libattr.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libattr.so) \
/usr/lib/libattr.so
```

- ¿Para qué sirve este paquete? Incluya una breve explicación de qué son los *extended attributes* en un sistema de archivos.

gg. Instalación de Acl-2.2.52 (0.1 SBUs):

1. Aplique el mismo cambio de versionamiento que se aplicó al paquete anterior:

```
sed -i -e 's|/@pkg_name@|&-@pkg_version@|' \
include/builddefs.in
```

2. Ejecute las siguientes instrucciones para aplicar correcciones a algunos *tests* y al paquete en sí:

```
sed -i "s:| sed.*::g" test/{sbits-restore,cp,misc}.test
sed -i -e "/TABS-1;/a if (x > (TABS-1)) x = (TABS-1);" \
libacl/__acl_to_any_text.c
```

3. Configure para compilar, compile, instale; y mueva la biblioteca compartida como en el paquete anterior:

```
./configure --prefix=/usr \
--bindir=/bin \
--disable-static \
--libexecdir=/usr/lib

make
make install install-dev install-lib
chmod -v 755 /usr/lib/libacl.so

mv -v /usr/lib/libacl.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libacl.so) \
/usr/lib/libacl.so
```

- ¿Qué son las listas de control de acceso (ACL's)?

hh. Instalación de Libcap-2.25 (0.1 SBUs):

1. Nuevamente evitamos la instalación de una *static library* que no se necesitará:

```
sed -i '/install.*STALIBNAME/d' libcap/Makefile
```

2. Compile, instale y mueva la biblioteca compartida como en los paquetes anteriores, usando las siguientes instrucciones:

```
make
```

```
make RAISE_SETFCAP=no lib=lib prefix=/usr install
chmod -v 755 /usr/lib/libcap.so

mv -v /usr/lib/libcap.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libcap.so) \
/usr/lib/libcap.so
```

- Explique la relación entre listas de control de acceso y *capabilities*.

ii. Instalación de Sed-4.4 (0.3 SBUs):

1. Ejecute los siguientes comandos para corregir un problema con el ambiente del LFS y un *test* que falla:

```
sed -i 's/usr/tools/' build-aux/help2man
sed -i 's/testsuite.panic-tests.sh//' Makefile.in
```

2. Ejecute el siguiente comando de configuración:

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

3. Compile el paquete y su documentación:

```
make
make html
```

4. Ejecute la *test suite*:

```
make check
```

5. Si todo salió bien, instale el paquete y su documentación:

```
make install
install -d -m755 /usr/share/doc/sed-4.4
install -m644 doc/sed.html /usr/share/doc/sed-4.4
```

jj. Instalación de Shadow-4.5 (0.2 SBUs):

1. Ejecute las siguientes instrucciones para deshabilitar la instalación del programa `groups` (Coreutils incluye una mejor versión) y su documentación:

```
sed -i 's/groups$(EXEEXT) //' src/Makefile.in
find man -name Makefile.in -exec sed -i 's/groups\.1 / /' {} \;
find man -name Makefile.in -exec sed -i 's/getspnam\.3 / /' {} \;
find man -name Makefile.in -exec sed -i 's/passwd\.5 / /' {} \;
```

- ¿Para qué sirve Shadow?

2. Ejecute los siguientes comandos para cambiar la metodología de cifrado usada por Shadow y la ubicación de los buzones de correo que trae por defecto:

```
sed -i -e 's@#ENCRYPT_METHOD DES@ENCRYPT_METHOD SHA512@' \
-e 's@/var/spool/mail@/var/mail@' etc/login.defs
```

   - ¿Qué son `/var/spool/mail` y `/var/mail`; y cuál es su relación con Shadow?

3. Ejecute la siguiente instrucción para acoplar la configuración de `useradd` al archivo `groups` creado anteriormente:

```
sed -i 's/1000/999/' etc/useradd
```

4. Ejecute el comando de configuración:

```
./configure --sysconfdir=/etc \
--with-group-name-max-length=32
```

5. Compile e instale:

```
make
make install
```

6. Mueva el archivo `passwd` a su correcta ubicación con la siguiente instrucción:

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

7. Habilite el sombreo (*shadowing*) de contraseñas para usuarios y grupos con las siguientes instrucciones:

```
pwconv
grpconv
```

8. Establezca el *password* para `root` con la siguiente instrucción (¡no lo olvide! Apúntelo si es necesario):

```
passwd root
```

kk. Instalación de Psmisc-23.1 (0.1 SBUs):

1. Configure, compile e instale con los siguientes comandos:

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

2. Mueva los programas `fuser` y `killall` a la ubicación estándar:

```
mv -v /usr/bin/fuser /bin
mv -v /usr/bin/killall /bin
```

ll. Instalación de Iana-Etc-2.30 (0.1 SBUs):

1. "Compile" e instale con `make` y `make install`, respectivamente.

mm. Instalación de Bison-3.0.4 (0.3 SBUs):

1. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr --docdir=/usr/share/doc/bison-3.0.4
```

2. Compile con `make`.
3. Instale con `make install`.

nn. Instalación de Flex-2.6.4 (0.4 SBUs):

1. Corrija un problema ocasionado por glibc-2.26, con la siguiente instrucción:

```
sed -i "/math.h/a #include <malloc.h>" src/flexdef.h
```

2. Con el siguiente comando, miéntale al proceso de configuración que sí está instalado el paquete `HELP2MAN`:

```
HELP2MAN=/tools/bin/true \
./configure --prefix=/usr \
--docdir=/usr/share/doc/flex-2.6.4
```

3. Compile con `make` y ejecute la *test suite* con `make check`.
4. Instale con `make install`.
5. Cree un *link* simbólico para los programas que dependen de Lex:

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

- ¿Cuál es la diferencia entre Lex y Flex?

oo. Instalación de Grep-3.1 (0.4 SBUs):

1. Ejecute el siguiente comando de configuración:

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

2. Compile con `make` y ejecute la *test suite* con `make check`.
3. Si todo sale bien, instale con `make install`.

pp. Instalación de Bash-4.4 (2.0 SBUs):

1. Aplique un parche necesario con la siguiente instrucción:

```
patch -Np1 -i ../bash-4.4-upstream_fixes-1.patch
```

2. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr \
--docdir=/usr/share/doc/bash-4.4 \
--without-bash-malloc \
--with-installed-readline
```

3. Compile con `make`.
4. Cambie la propiedad de la carpeta y su contenido a `nobody`, y ejecute la *test suite* con ese usuario usando el siguiente comando (usar `nobody` permite la realización de más pruebas):

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make tests"
```

5. Instale con `make install` y mueva el archivo ejecutable instalado a la carpeta `/bin` con `mv -vf /usr/bin/bash /bin`.
6. Reemplace la *shell* que está usando por la recién instalada:

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

qq. Instalación de Libtool-2.4.6 (1.8 SBUs):

1. Configure con `./configure --prefix=/usr`.
2. Compile con `make` y realice los *tests* con `make check` (toma un buen tiempo).
3. Si todo está bien (a excepción de cinco *tests* que fallan en LFS), instale con `make install`.

- ¿Cuál es el propósito/beneficio que provee el *GNU library support script*?

Construcción de LFS (parte II)

a. Instalación de GDBM-1.13 (0.1 SBUs):
   i. Ejecute la configuración con el siguiente comando:

```
./configure --prefix=/usr \
--disable-static \
--enable-libgdbm-compat
```

   ii. Compile con `make` y ejecute la *test suite* con `make check`.
   iii. Si todo está bien, instale con `make install`.

b. Instalación de Gperf-3.1 (0.1 SBUs):
   i. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr --docdir=/usr/share/doc/gperf-3.1
```

ii. Compile con `make` y ejecute la *test suite* con `make -j1 check`.
iii. Si todo sale bien, instale con `make install`

- ¿Qué es un *perfect hash*?

c. Instalación de Expat-2.2.3 (0.1 SBUs):
   i. Aplique la siguiente corrección a uno de los *tests*:

```
sed -i 's|usr/bin/env |bin/|' run.sh.in
```

   ii. Ejecute la configuración con el siguiente comando:

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

   iii. Compile con `make` y ejecute la *test suite* con `make check`.
   iv. Si todo sale bien, instale con `make install`.
   v. Instale la documentación con los siguientes comandos:

```
install -v -dm755 /usr/share/doc/expat-2.2.3
install -v -m644 doc/*.{html,png,css} \
/usr/share/doc/expat-2.2.3
```

d. Instalación de Inetutils-1.9.4 (0.3 SBUs):
   i. Ejecute la siguiente instrucción de configuración:

```
./configure --prefix=/usr \
--localstatedir=/var \
--disable-logger \
--disable-whois \
--disable-rcp \
--disable-rexec \
--disable-rlogin \
--disable-rsh \
--disable-servers
```

   ii. Compile con `make` y ejecute la *test suite* con `make check`.
   iii. Si todo sale bien instale con `make install`.
   iv. Con el siguiente comando reubique algunos de los programas instalados al directorio `/sbin`, en caso de que su directorio actual de instalación no esté disponible en algún momento:

```
mv -v /usr/bin/{hostname,ping,ping6,traceroute} /bin
mv -v /usr/bin/ifconfig /sbin
```

- ¿Cuál es el propósito del directorio `sbin`?

Instalación de Perl

e. Instalación de Perl-5.26.0 (8.6 SBUs):

i. Cree el archivo `/etc/hosts`, necesario para la configuración de Perl y para sus *tests*:

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

- ¿Cuál es el propósito del archivo `/etc/hosts`?

ii. Indique a Perl que use las versiones de Zlib y Bzip2 instaladas en nuestro sistema (en lugar de sus versiones internas) con los siguientes comandos:

```
export BUILD_ZLIB=False
export BUILD_BZIP2=0
```

iii. Ejecute el siguiente comando de configuración:

```
sh Configure -des -Dprefix=/usr \
-Dvendorprefix=/usr \
-Dman1dir=/usr/share/man/man1 \
-Dman3dir=/usr/share/man/man3 \
-Dpager="/usr/bin/less -isR" \
-Duseshrplib \
-Dusethreads
```

- ¿Qué hacen las opciones `-des` y qué diferencia habría si no se usan?
- El libro indica que la opción `-Dpager` de arriba sirve para usar `less` en lugar de `more`. ¿Qué es el *pager* de Perl?

iv. Compile con `make` y ejecute la *test suite* con `make -k test`, esperando que fallen algunos *tests* relacionados con Zlib.

v. Si todo sale bien, instale con `make install` y no olvide eliminar las variables declaradas anteriormente:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

f. Instalación de XML::Parser-2.44 (0.1 SBUs):

i. El paquete se descargó como `XML-Parser-2.44.tar.gz`. Una vez que lo haya descomprimido, ejecute el siguiente comando de configuración:

```
perl Makefile.PL
```

ii. Compile con `make` y ejecute la *test suite* con `make test`.
iii. Si todo sale bien, instale con `make install`.

g. Instalación de Intltool-0.51.0 (0.1 SBUs):
   i. Use el siguiente comando para evitar una advertencia ocasionada por Perl:

```
sed -i 's:\\\${:\\\$\\{:' intltool-update.in
```

   ii. Configure el paquete con `./configure --prefix=/usr`.
   iii. Compile con `make` y ejecute la *test suite* con `make check`.
   iv. Si todo sale bien ejecute las siguientes instrucciones para instalar el paquete y su documentación:

```
make install
install -v -Dm644 doc/I18N-HOWTO \
/usr/share/doc/intltool-0.51.0/I18N-HOWTO
```

h. Instalación de Autoconf-2.69 (3.3 SBUs):
   i. Configure la instalación con `./configure --prefix=/usr`.
   ii. Compile con `make` y ejecute la *test suite* con `make check` (esto es lo que toma más tiempo, pero no es demasiado).
   iii. Si todo sale bien (a excepción un par de *tests* que fallan) instale con `make install`.

Instalación de Automake

i. Instalación de Automake-1.15.1 (8.5 SBUs):
   i. Configure con la siguiente instrucción:

```
./configure --prefix=/usr \
--docdir=/usr/share/doc/automake-1.15.1
```

   ii. Compile con `make`.
   iii. Modifique un archivo para que defina una variable de ambiente antes de ejecutar un *test*, con el propósito de que se use la correcta biblioteca de Flex. Use el siguiente comando:

```
sed -i "s:./configure:LEXLIB=/usr/lib/libfl.a &:" t/lex-
{clean,depend}-cxx.sh
```

   iv. Ejecute la *test suite* con el siguiente comando (toma un buen tiempo):

```
make -j4 check
```

   v. Si todo sale bien (a excepción de dos *tests* que fallan), instale con `make install`.
      - Describa el propósito de, y la relación entre, Automake y Autoconf.

j. Instalación de Xz-5.2.3 (0.2 SBUs):
   i. Ejecute la configuración con el siguiente comando:

```
./configure --prefix=/usr \
--disable-static \
--docdir=/usr/share/doc/xz-5.2.3
```

   ii. Compile con `make` y ejecute la *test suite* con `make check`.
   iii. Si todo sale bien, instale con `make install` y realice las siguientes reubicaciones necesarias:

```
mv -v /usr/bin/{lzma,unlzma,lzcat,xz,unxz,xzcat} /bin
mv -v /usr/lib/liblzma.so.* /lib
ln -svf ../../lib/$(readlink /usr/lib/liblzma.so) \
/usr/lib/liblzma.so
```

k. Instalación de Kmod-24 (0.1 SBUs):
   i. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr \
--bindir=/bin \
--sysconfdir=/etc \
--with-rootlibdir=/lib \
--with-xz \
--with-zlib
```

   ii. Compile con `make`.
   iii. Instale con `make install` y use las siguientes instrucciones para crear *links* simbólicos que proveen compatibilidad con un antiguo paquete para manejo de módulos de *kernel*:

```
for target in depmod insmod lsmod modinfo modprobe \
rmmod; do
     ln -sv ../bin/kmod /sbin/$target
done

ln -sv kmod /bin/lsmod
```

l. Instalación de Gettext-0.19.8.1 (2.4 SBUs):
   i. Ejecute las siguientes instrucciones para evitar unos comandos que podrían ocasionar un ciclo infinito:

```
sed -i '/^TESTS =/d' gettext-runtime/tests/Makefile.in &&
sed    -i    's/test-lock..EXEEXT.//'    gettext-tools/gnulib-
tests/Makefile.in
```

ii. Configure el paquete con la siguiente instrucción:

```
./configure --prefix=/usr \
--docdir=/usr/share/doc/gettext-0.19.8.1
```

iii. Compile con `make` y ejecute la *test suite* con `make check` (toma un tiempo).
iv. Si todo sale bien, instale con `make install` y modifique los permisos de una biblioteca instalada mediante la siguiente instrucción:

```
chmod -v 0755 /usr/lib/preloadable_libintl.so
```

m. Instalación de Procps-ng-3.3.12 (0.1 SBUs):
i. Ejecute la siguiente instrucción de configuración:

```
./configure --prefix=/usr \
--exec-prefix= \
--libdir=/usr/lib \
--docdir=/usr/share/doc/procps-ng-3.3.12 \
--disable-static \
--disable-kill
```

ii. Compile con `make`.
iii. Modifique la *test suite* para evitar una falla y corregir dos pruebas, con las siguientes instrucciones:

```
sed -i -r 's|(pmap_initname)\\\$|\1|' \
testsuite/pmap.test/pmap.exp

sed -i '/set tty/d' testsuite/pkill.test/pkill.exp
rm testsuite/pgrep.test/pgrep.exp
```

- ¿Qué es un dispositivo `tty`?

iv. Ejecute la *test suite* con `make check` (fallará un *test*).
v. Instale con `make install`.
vi. Concluya realizando las siguientes reubicaciones necesarias:

```
mv -v /usr/lib/libprocps.so.* /lib
ln -sfv ../../lib/$(readlink /usr/lib/libprocps.so) \
/usr/lib/libprocps.so
```

Instalación de E2fsprogs

n. Instalación de E2fsprogs-1.43.5 (3.3 SBUs):

i. Cree y acceda un directorio llamado `build`.
ii. Ejecute el siguiente comando de configuración:

```
LIBS=-L/tools/lib \
CFLAGS=-I/tools/include \
PKG_CONFIG_PATH=/tools/lib/pkgconfig \
../configure --prefix=/usr \
--bindir=/bin \
--with-root-prefix="" \
--enable-elf-shlibs \
--disable-libblkid \
--disable-libuuid \
--disable-uuidd \
--disable-fsck
```

iii. Compile con `make`.
iv. Cree unos *links* simbólicos necesarios para la correcta ejecución de los *tests*, y ejecute los *tests* (fallarán tres *tests* relacionados con la característica "quota", debido a un *bug*):

```
ln -sfv /tools/lib/lib{blk,uu}id.so.1 lib
make LD_LIBRARY_PATH=/tools/lib check
```

v. Instale con `make install`.
vi. Instale las bibliotecas estáticas con `make install-libs`.
vii. Para luego poder remover los símbolos de *debugging* como se hizo anteriormente, modifique los permisos de estas bibliotecas estáticas con la siguiente instrucción:

```
chmod -v u+w /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

viii. Extraiga un archivo de documentación instalado y actualice el archivo `dir` con las siguientes instrucciones.:

```
gunzip -v /usr/share/info/libext2fs.info.gz
install-info --dir-file=/usr/share/info/dir \
/usr/share/info/libext2fs.info
```

- ¿Para qué sirve el archivo `dir` ubicado en `/usr/share/info`?

ix. Instale documentación adicional con los siguientes comandos:

```
makeinfo -o doc/com_err.info ../lib/et/com_err.texinfo
install -v -m644 doc/com_err.info /usr/share/info
install-info --dir-file=/usr/share/info/dir \
/usr/share/info/com_err.info
```

- ¿Cuál es la diferencia entre un sistema de archivos común y un sistema de archivos que lleva un diario (*journaling file system*)?

o. Instalación de Coreutils-8.27 (2.4 SBUs):

i. Aplique un parche que corrige errores relacionados con internacionalización en los programas de este paquete:

```
patch -Np1 -i ../coreutils-8.27-i18n-1.patch
```

ii. Aplique una corrección que evita la posibilidad de un ciclo infinito:

```
sed -i '/test.lock/s/^/#/' gnulib-tests/gnulib.mk
```

iii. Ejecute la configuración del paquete con la siguiente instrucción (es importante que use la variable `FORCE_UNSAFE_CONFIGURE` como se indica, porque de lo contrario el paquete impide la configuración como `root`):

```
FORCE_UNSAFE_CONFIGURE=1 ./configure \
--prefix=/usr \
--enable-no-install-program=kill,uptime
```

iv. Compile con `FORCE_UNSAFE_CONFIGURE=1 make.`

v. Ejecute el siguiente comando para realizar *tests* que requieren el usuario `root`:

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

vi. Los siguientes *tests* se realizarán con el usuario `nobody`. Para ello, agregue este usuario a un grupo nuevo en el archivo `/etc/group` con la siguiente instrucción:

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

vii. Haga a `nobody` el propietario de todo lo contenido en la carpeta actual para poder realizar los *tests*:

```
chown -Rv nobody .
```

viii. Ejecute estos *tests* de forma no interactiva, y con el usuario `nobody`, por medio de este comando:

```
su nobody -s /bin/bash \
-c "PATH=$PATH make RUN_EXPENSIVE_TESTS=yes check"
```

ix. Fallará un *test* llamado `date-debug` pero, si todo lo demás salió bien, elimine el grupo creado para hacer los *tests*:

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

x. Instale con `make install`.

xi. Ejecute los siguientes comandos para reubicar varios programas a sus directorios estándar:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} \
/bin
mv -v /usr/bin/{false,ln,ls,mkdir,mknod,mv,pwd,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 \
/usr/share/man/man8/chroot.8
sed -i s/\"1\"/\"8\"/1 /usr/share/man/man8/chroot.8
mv -v /usr/bin/{head,sleep,nice,test,[} /bin
```

p. Instalación de Diffutils-3.6 (0.4 SBUs):
   i. Configure con `./configure --prefix=/usr`.
   ii. Compile con `make` y ejecute la *test suite* con `make check`.
   iii. Si todo está bien, instale con `make install`.

q. Instalación de Gawk-4.1.4 (0.3 SBUs):
   i. Configure el paquete con `./configure --prefix=/usr`.
   ii. Compile con `make` y ejecute la *test suite* con `make check`.
   iii. Si todos los *tests* pasan, instale con `make install`.
   iv. Instale la documentación con los siguientes comandos:

```
mkdir -v /usr/share/doc/gawk-4.1.4
cp -v doc/{awkforai.txt,*.{eps,pdf,jpg}} \
/usr/share/doc/gawk-4.1.4
```

- Gawk es la implementación GNU de Awk. Describa Awk.

r. Instalación de Findutils-4.6.0 (0.7 SBUs):
   i. Elimine un *test* que puede ocasionar un ciclo infinito, usando el siguiente comando:

```
sed -i 's/test-lock..EXEEXT.//' tests/Makefile.in
```

   ii. Configure el paquete con la siguiente instrucción:

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

   iii. Compile con `make` y ejecute la *test suite* con `make check`.
   iv. Si todo está bien, instale con `make install`.

v. Los siguientes comandos cambian la ubicación del programa instalado `find` y corrigen un problema con el *script* encargado de actualizar la base de datos de archivos en el sistema:

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

s. Instalación de Groff-1.22.3 (0.4 SBUs):
   1. Configure el paquete con el siguiente comando:

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

      - ¿Cuál es la importancia de Groff (y sus predecesores) en el contexto de los sistemas Unix y sus derivados?
   2. Compile con `make -j1`, e instale con `make install`.

t. Instalación de GRUB-2.02 (0.8 SBUs):
   1. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr \
--sbindir=/sbin \
--sysconfdir=/etc \
--disable-efiemu \
--disable-werror
```

   2. Compile con `make` e instale con `make install`.

u. Instalación de Less-487 (0.1 SBUs):
   1. Ejecute el siguiente comando de configuración:

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

   2. Compile con `make` e instale con `make install`.

v. Instalación de Gzip-1.8 (0.1 SBUs):
   1. Configure el paquete con `./configure --prefix=/usr`.
   2. Compile con `make` y ejecute la *test suite* con `make check` (fallará un *test* llamado "help-version").
   3. Si todo sale bien, instale con `make install` y realice la siguiente reubicación necesaria:

```
mv -v /usr/bin/gzip /bin
```

w. Instalación de IPRoute2-4.12.0 (0.2 SBUs):

1. Ejecute los siguientes comandos para evitar la instalación de documentación del programa `arpd`, que no forma parte del LFS:

```
sed -i /ARPD/d Makefile
sed -i 's/arpd.8//' man/man8/Makefile
rm -v doc/arpd.sgml
```

2. Ejecute el siguiente comando para evitar compilar un módulo que requiere otro paquete no instalado en este LFS:

```
sed -i 's/m_ipt.o//' tc/Makefile
```

3. Compile con `make`.
4. Instale con el siguiente comando:

```
make DOCDIR=/usr/share/doc/iproute2-4.12.0 install
```

x. Instalación de Kbd-2.0.4 (0.1 SBUs):

1. Aplique un parche para lograr comportamiento consistente de este paquete con las teclas *delete* y *backspace*:

```
patch -Np1 -i ../kbd-2.0.4-backspace-1.patch
```

2. Remueva el programa y la documentación de `resizecons` de la instalación por medio de los siguientes comandos, ya que sus funcionalidades son provistas por otro programa:

```
sed -i 's/\(RESIZECONS_PROGS=\)yes/\1no/g' configure
sed -i 's/resizecons.8 //' docs/man/man8/Makefile.in
```

3. Configure el paquete con el siguiente comando:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure \
--prefix=/usr \
--disable-vlock
```

4. Compile con `make` y ejecute la *test suite* con `make check`.
5. Si todo sale bien, instale con `make install`.
6. Instale la documentación con las siguientes instrucciones:

```
mkdir -v /usr/share/doc/kbd-2.0.4
cp -R -v docs/doc/* /usr/share/doc/kbd-2.0.4
```

y. Instalación de Libpipeline-1.4.2 (0.1 SBUs):

1. Ejecute el siguiente comando de configuración:

```
PKG_CONFIG_PATH=/tools/lib/pkgconfig ./configure \
--prefix=/usr
```

2. Compile con `make` y ejecute la *test suite* con `make check`
3. Si todo sale bien, instale con `make install`

z. Instalación de Make-4.2.1 (0.6 SBUs):

1. Configure el paquete con `./configure --prefix=/usr`.
2. Compile con `make`.
3. Ejecute la *test suite* con la siguiente instrucción:

```
make PERL5LIB=$PWD/tests/ check
```

4. Si todo sale bien, instale con `make install`.

aa. Instalación de Patch-2.7.5 (0.2 SBUs):

1. Configure el paquete con `./configure --prefix=/usr`.
2. Compile con `make` y ejecute la *test suite* con `make check`.
3. Si todo sale bien, instale con `make install`.

bb. Instalación de Sysklogd-1.5.1 (0.1 SBUs):

1. Ejecute los siguientes comandos para corregir un problema que ocasiona una *segmentation fault* en `klogd` y corregir el uso de una característica obsoleta:

```
sed -i '/Error loading kernel symbols/{n;n;d}' ksym_mod.c
sed -i 's/union wait/int/' syslogd.c
```

2. Compile con `make` e instale con `make BINDIR=/sbin install`.
3. Cree un archivo de configuración para el paquete:

```
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
```

- Explique el contenido y propósito de este archivo de configuración.

cc. Instalación de Sysvinit-2.88dsf (0.1 SBUs):

1. Aplique un parche necesario para evitar programas instalados por otros paquetes, y realizar otras correcciones:

```
patch -Np1 -i ../sysvinit-2.88dsf-consolidated-1.patch
```

2. Compile con `make -C src`.
3. Instale con `make -C src install`.

- Explique los *run levels* de SystemV.
- Investigue otro *init system*, y explique brevemente las diferencias con SysV Init.

Instalación de Eudev

dd. Instalación de Eudev-3.2.2 (0.2 SBUs):

1. Ejecute el siguiente comando para prevenir un error en la *test suite*:

```
sed -r -i 's|/usr(/bin/test)|\1|' test/udev-test.pl
```

2. Use la siguiente instrucción para evitar un error en la compilación:

```
sed -i '/keyboard_lookup_key/d' src/udev/udev-builtin-keyboard.c
```

3. Cree un archivo que evitará usar un directorio de la *toolchain* en los ejecutable de este paquete:

```
cat > config.cache << "EOF"
HAVE_BLKID=1
BLKID_LIBS="-lblkid"
BLKID_CFLAGS="-I/tools/include"
EOF
```

4. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr \
--bindir=/sbin \
--sbindir=/sbin \
--libdir=/usr/lib \
--sysconfdir=/etc \
--libexecdir=/lib \
--with-rootprefix= \
--with-rootlibdir=/lib \
--enable-manpages \
--disable-static \
--config-cache
```

5. Compile con `LIBRARY_PATH=/tools/lib make`.
6. Cree los siguientes directorios necesarios para la instalación y las pruebas:

```
mkdir -pv /lib/udev/rules.d
mkdir -pv /etc/udev/rules.d
```

7. Ejecute la *test suite* con `make LD_LIBRARY_PATH=/tools/lib check`.
8. Si todo sale bien, instale con `make LD_LIBRARY_PATH=/tools/lib install`.
9. Ejecute las siguientes instrucciones para configurar el paquete:

```
tar -xvf ../udev-lfs-20140408.tar.bz2
make -f udev-lfs-20140408/Makefile.lfs install
LD_LIBRARY_PATH=/tools/lib udevadm hwdb --update
```

- Explique brevemente para qué sirve y por qué es importante este paquete. ¿Cuál es la relación entre `udev` y `systemd`?
- Las primeras instrucciones instalan algunas reglas y archivos auxiliares necesarios. ¿Qué son "reglas" y cuál es el propósito de las que se instalan?
- La última instrucción crea la base de datos inicial de dispositivos de *hardware* en el archivo `/etc/udev/hwdb.bin`. ¿Qué información almacena este archivo?

ee. Instalación de Util-linux-2.30.1 (1.0 SBUs):

1. Cree un directorio donde almacenar el archivo `adjtime` y el programa `hwclock`, de acuerdo con el estándar FHS:

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

- ¿Qué hacen el archivo de configuración `adjtime` y el programa `hwclock`?

2. Ejecute el siguiente comando de configuración:

```
./configure ADJTIME_PATH=/var/lib/hwclock/adjtime \
--docdir=/usr/share/doc/util-linux-2.30.1 \
--disable-chfn-chsh \
--disable-login \
--disable-nologin \
--disable-su \
--disable-setpriv \
--disable-runuser \
--disable-pylibmount \
--disable-static \
--without-python \
--without-systemd \
--without-systemdsystemunitdir
```

3. Compile con `make`.
4. Otorgue propiedad de la carpeta y su contenido al usuario `nobody` para la ejecución de la *test suite*, y ejecute la *test suite* con este usuario:

```
chown -Rv nobody .
su nobody -s /bin/bash -c "PATH=$PATH make -k check"
```

   - El libro indica que para poder ejecutar los *tests* como `root` es necesario que el sistema en el que se está trabajando tenga habilitada como módulo la opción `CONFIG_SCSI_DEBUG`. ¿Qué son `SCSI` y los `scsi_debug` *devices*?

5. Si todos los *tests* pasan, instale con `make install`.

   - Liste al menos tres programas de uso común que son instalados por este paquete. Puede incluir los que hemos usado a lo largo del proyecto y en laboratorios.

ff. Instalación de Man-DB-2.7.6.1 (0.4 SBUs):

1. Ejecute el siguiente comando de configuración:

```
./configure --prefix=/usr \
--docdir=/usr/share/doc/man-db-2.7.6.1 \
--sysconfdir=/etc \
--disable-setuid \
--enable-cache-owner=bin \
--with-browser=/usr/bin/lynx \
--with-vgrind=/usr/bin/vgrind \
--with-grap=/usr/bin/grap \
--with-systemdtmpfilesdir=
```

2. Compile con `make` y ejecute la *test suite* con `make check`.
3. Si todo sale bien, instale con `make install`

gg. Instalación de Tar-1.29 (2.6 SBUs):

1. Ejecute el siguiente comando de configuración:

```
FORCE_UNSAFE_CONFIGURE=1 \
./configure --prefix=/usr --bindir=/bin
```

2. Compile con `make`.
3. Ejecute las siguientes instrucciones para instalar el paquete y su documentación:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.29
```

hh. Instalación de Texinfo-6.4 (1.1 SBUs):

1. Configure el paquete con `./configure --prefix=/usr --disable-static`.
2. Compile con `make` y ejecute la *test suite* con `make check`.

3. Si todo sale bien, instale con `make install`.

- Explique brevemente la diferencia entre *man pages* e *info pages*.

ii. Instalación de Vim-8.0.586 (1.1 SBU):

1. Ejecute la siguiente instrucción para modificar la ubicación donde se instalará `vimrc`:

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

- ¿Qué es `vimrc`?

2. Use el siguiente comando para evitar un *test* que falla:

```
sed -i '/call/{s/split/xsplit/;s/303/492/}' src/testdir/test_recover.vim
```

3. Configure el paquete con `./configure --prefix=/usr`.
4. Compile con `make`.
5. Ejecute la *test suite*, dirigiendo su *output* a un archivo con el siguiente comando:

```
make -j1 test &> vim-test.log
```

6. Revise `vim-test.log`. Si cerca del final del archivo encuentra las palabras `ALL DONE`, instale el paquete con `make install`.
7. Vim es una versión renovada del editor de texto tradicional Vi. Ejecute las siguientes instrucciones para que el comando `vi` (y *man pages* asociadas) ejecute `vim`:

```
ln -sv vim /usr/bin/vi
for L in /usr/share/man/{,*/}man1/vim.1; do
     ln -sv vim.1 $(dirname $L)/vi.1
done
```

- ¿Qué ventajas presenta Vim sobre Vi?

8. Agregue el siguiente *link* simbólico para que la ubicación de las *man pages* de Vim sea consistente con la de las *man pages* de los demás paquetes:

```
ln -sv ../vim/vim80/doc /usr/share/doc/vim-8.0.586
```

9. Cree un archivo de configuración con las siguientes instrucciones:

```
cat > /etc/vimrc << "EOF"
" Begin /etc/vimrc

set nocompatible
set backspace=2
set mouse=r
```

```
syntax on
if (&term == "xterm") || (&term == "putty")
      set background=dark
endif

" End /etc/vimrc
EOF
```

10. Ejecute el comando `touch ~/.vimrc`.

Últimos pasos

jj. No vamos a remover los *debugging symbols* esta vez porque el ahorro en espacio no es significativo, y corremos el riesgo de arruinar nuestro sistema al hacerlo. Únicamente limpiaremos algunos archivos creados durante los *tests* con el siguiente comando:

```
rm -rf /tmp/*
```

Y reingresaremos al ambiente `chroot` con los siguientes comandos:

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

Tome en cuenta que lo anterior (exceptuando `logout`) será lo que debe ejecutar para continuar desarrollando LFS si por alguna razón sale del ambiente `chroot`. Si apaga o reinicia su máquina virtual no olvide que también deberá volver a montar los sistemas de archivos ANTES de volver a entrar al ambiente `chroot`. No eliminaremos el directorio `/tools`.

Finalmente, completaremos el LFS:

a. Instalación de los *bootscripts*:

Extraiga los *bootscripts* provistos con LFS en su carpeta *sources* (`lfs-bootscripts-20170626.tar.bz`) y acceda a la carpeta resultante. Luego ejecute el comando `make install`.

b. Ejecute un *script* para modificación de las reglas de nombramiento de Udev para dispositivos de red, con el siguiente comando:

```
bash /lib/udev/init-net-rules.sh
```

c. Revise el contenido del archivo `/etc/udev/rules.d/70-persistent-net.rules`.

- ¿Cuál es el nombre de los dispositivos de red? ¿Qué significan los demás datos?

d. Diríjase al directorio `/etc/sysconfig` y cree un archivo llamado `ifconfig.eth0` con el siguiente comando:

```
cd /etc/sysconfig/

cat > ifconfig.eth0 << "EOF"

ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255

EOF
```

- ¿Para qué sirve este archivo en el directorio `/etc/sysconfig`?

e. Cree el archivo de configuración de DNS's con el siguiente comando:

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

nameserver 8.8.8.8
nameserver 8.8.4.4

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

f. Provea un *hostname* a su sistema con el siguiente comando:

```
echo "ELEEFEESE" > /etc/hostname
```

g. Finalmente, cree el archivo `/etc/hosts`:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts

127.0.0.1 localhost.localdomain localhost
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

# End /etc/hosts
EOF
```

h. Creación del archivo `inittab`:

Cree un archivo en el directorio `/etc` llamado `inittab`. Debe contener lo siguiente:

```
# Begin /etc/inittab

id:3:initdefault:

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

l0:0:wait:/etc/rc.d/init.d/rc 0
l1:S1:wait:/etc/rc.d/init.d/rc 1
l2:2:wait:/etc/rc.d/init.d/rc 2
l3:3:wait:/etc/rc.d/init.d/rc 3
l4:4:wait:/etc/rc.d/init.d/rc 4
l5:5:wait:/etc/rc.d/init.d/rc 5
l6:6:wait:/etc/rc.d/init.d/rc 6

ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

su:S016:once:/sbin/sulogin

1:2345:respawn:/sbin/agetty --noclear tty1 9600
2:2345:respawn:/sbin/agetty tty2 9600
3:2345:respawn:/sbin/agetty tty3 9600
4:2345:respawn:/sbin/agetty tty4 9600
5:2345:respawn:/sbin/agetty tty5 9600
6:2345:respawn:/sbin/agetty tty6 9600

# End /etc/inittab
```

- ¿Cómo funcionan los *run levels* de `init`?

i. Configuración del reloj de sistema:

Cree un archivo en el directorio `/etc/sysconfig` llamado `clock`. El contenido debe ser el siguiente:

```
# Begin /etc/sysconfig/clock

UTC=0

# Set this to any options you might need to give to hwclock,
# such as machine hardware clock type for Alphas.
CLOCKPARAMS=

# End /etc/sysconfig/clock
```

j. Despliegue de revisión del sistema de archivos durante el *booteo*:

Abra el archivo `/etc/sysconfig/rc.site` y busque la variable `VERBOSE_FSCK`. Modifique su valor a '`y`', y habilite la variable eliminando el caracter de comentarios (*i.e.,* '`#`').

k. Configuración de la *shell*:

Ejecute las siguientes instrucciones y compare su resultado con los desplegados abajo de ellas para comprobar que el `locale en_US` es soportado por Glibc:

```
LC_ALL=en_US locale language
LC_ALL=en_US locale charmap
LC_ALL=en_US locale int_curr_symbol
LC_ALL=en_US locale int_prefix
```

```
English
ISO-8859-1
USD
1
```

Cree un archivo en el directorio `/etc` llamado `profile`. Este archivo debe contener lo siguiente (basado en los resultados de los comandos anteriores):

```
# Begin /etc/profile

export LANG=en_US.ISO-8859-1

# End /etc/profile
```

Si los resultados de las pruebas con `LC_ALL` no son los esperados deberá buscar otro `locale` que despliegue correctamente su idioma, conjunto de caracteres, moneda y prefijo telefónico para usar en `/etc/profile`. Puede consultar los `locales` soportados mediante el comando `locale -a`.

I. Configuración de teclado:

Cree un archivo en el directorio `/etc` llamado `inputrc`. Su contenido debe ser el siguiente:

```
# 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
```

- ¿Cuál es la importancia del paquete Readline?

m. Configuración de *shells* válidas:

Cree un archivo en el directorio `/etc` llamado `shells`. Su contenido debe ser el siguiente:

```
# Begin /etc/shells

/bin/sh
/bin/bash

# End /etc/shells
```

n. Creación de la tabla de sistemas de archivos:

Cree un archivo en el directorio `/etc` llamado `fstab`. Su contenido debe ser el siguiente:

```
# Begin /etc/fstab

# file system    mountpoint type       options               dump
# fsck order

/dev/sda2        /          ext4       defaults              1 1
/dev/sda3        swap       swap       pri=1                 0 0
proc             /proc      proc       nosuid,noexec,nodev   0 0
sysfs            /sys       sysfs      nosuid,noexec,nodev   0 0
devpts           /dev/pts   devpts     gid=5,mode=620        0 0
tmpfs            /run       tmpfs      defaults              0 0
devtmpfs         /dev       devtmpfs   mode=0755,nosuid      0 0

# End /etc/fstab
```

Este contenido supone que `/dev/sda2` es la partición donde se está desarrollando su sistema LFS; y que `/dev/sda3` es la partición de *swapping*. Si realizó los pasos de preparación de la máquina virtual siguiendo las instrucciones dadas, éste debería ser el caso.

Compilación e instalación de *kernel*

o. Instalación del *kernel*:

Diríjase al directorio `/sources` y extraiga el *kernel* de `linux-4.12.7.tar.xz`. Desde la carpeta donde lo extrajo, ejecute el comando `make mrproper`. Cree un archivo de configuración derivado de la configuración de su *host* con `make defconfig`. A continuación, ejecute el siguiente comando para desplegar el menú de configuración de *kernel*:

```
make LANG=en_US LC_ALL= menuconfig
```

Note el espacio luego de `LC_ALL=`. En el menú de configuración, diríjase al submenú `Device Drivers`, y en éste ingrese a `Generic Driver Options`. Deshabilite la opción `Support for uevent helper` y asegúrese de que la opción `Maintain a devtmpfs filesystem to mount at /dev` esté habilitada. Al terminar, grabe el archivo de configuración con el nombre `.config` y luego ejecute `make` (esto tomará un tiempo). Si la ejecución de `make` concluye sin problemas, proceda a ejecutar `make modules_install`.

Preparación de LFS para *booteo*

Luego de compilar exitosamente el *kernel* deberá desmontar la partición de *booteo* con el siguiente comando:

```
umount -v /boot
```

El directorio `/boot` seguirá existiendo, pero será parte de la partición en la que se desarrolló el LFS. Copie el *kernel*, el archivo de símbolos `System.map` y su archivo de configuración al directorio `/boot` con los siguientes comandos:

```
cp -v arch/i386/boot/bzImage /boot/vmlinuz-4.12.7-lfs-8.1
cp -v System.map /boot/System.map-4.12.7
cp -v .config /boot/config-4.12.7
```

Instale la documentación del *kernel* con las siguientes instrucciones:

```
install -d /usr/share/doc/linux-4.12.7
cp -r Documentation/* /usr/share/doc/linux-4.12.7
```

Salga del directorio a donde extrajo el *kernel* y modifique su propietario con `chown -R 0:0`. No se borrará el código fuente del *kernel*, por lo que es necesario asegurar que sólo `root` tenga permisos sobre él.
Proceda a crear un directorio en `/etc` llamado `modprobe.d` con el siguiente comando:

```
install -v -m755 -d /etc/modprobe.d
```

Finalmente, cree un archivo en el directorio `/etc/modprobe.d` llamado `usb.conf`. Su contenido debe ser el siguiente:

```
# Begin /etc/modprobe.d/usb.conf

install ohci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i ohci_hcd ; true
install uhci_hcd /sbin/modprobe ehci_hcd ; /sbin/modprobe -i uhci_hcd ; true

# End /etc/modprobe.d/usb.conf
```

p. Configuración de GRUB para *booteo* del LFS:
Vuelva a montar la partición de *booteo* con la siguiente instrucción:

```
mount -v -t ext2 /dev/sda1 /boot
```

Independiente de cómo ha manejado sus *snapshots* hasta ahora, asegúrese de crear un *snapshot* en este punto. Luego ejecute los siguientes comandos:

```
grub-install /dev/sda
grub-mkconfig -o /boot/grub/grub.cfg
```

Agregue, al final del archivo de configuración creado (`grub.cfg`), el siguiente código:

```
menuentry "Knoppix" {
      insmod reiserfs
      set root=(hd0,4)
      configfile /boot/grub/grub.cfg
}

menuentry "LFS" {
      insmod ext2
      set root=(hd0,2)
      configfile /boot/grub/grub.cfg
      echo 'Iniciando LFS Linux-4.12.7...'
      linux /boot/vmlinuz-4.12.7-lfs-8.1 root=/dev/sda2 ro single
}
```

Luego de esto, vuelva a desmontar la partición de *booteo* y ejecute los siguientes comandos:

```
grub-install /dev/sda2 --grub-setup=/bin/true
grub-mkconfig -o /boot/grub/grub.cfg
```

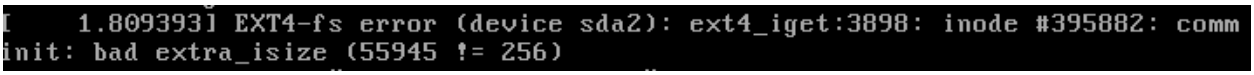
El comando anterior debería detectar el *kernel* compilado y agregar a `grub.cfg` (el recién creado en la partición del LFS) una opción para *booteo* de nuestro LFS.

Al terminar con esto saldrá del ambiente `chroot` y desmontará los sistemas virtuales de archivos montados para el LFS. Use las siguientes instrucciones:

```
logout
sudo umount -v $LFS/dev/pts
sudo umount -v $LFS/dev
sudo umount -v $LFS/run
sudo umount -v $LFS/proc
sudo umount -v $LFS/sys
sudo umount -v $LFS
```

Estando afuera del ambiente `chroot` ejecute la instrucción `sudo grub-mkconfig -o /boot/grub/grub.cfg`. Finalmente, reinicie su máquina virtual. GRUB debería iniciar mostrando las opciones `Knoppix` y `LFS`, y ambas deberían desplegar un submenú con una única opción que le llevará al sistema operativo correspondiente.

Es posible que su LFS presente un error durante el *booteo* similar a lo siguiente:



Esto evidencia problemas con el sistema de archivos instalado en la partición de LFS. Para arreglarlo, reinicie su máquina virtual y acceda a Knoppix. En Knoppix desmonte la partición de *booteo* y la de LFS, y ejecute un chequeo del sistema de archivos con el siguiente comando:

```
sudo fsck -y /dev/sda2
```

La opción `-y` asegura que se apliquen automáticamente todas las reparaciones sugeridas por el programa de chequeo. Esto no es recomendado, pero para este proyecto es efectivo. Una vez completada la revisión y las reparaciones deberá volver a montar los sistemas virtuales de archivos y acceder al ambiente `chroot` (recuerde que la instrucción para hacerlo cambió al concluir, con `Vim`, la instalación de programas del capítulo 6). Desde allí deberá reinstalar los *bootscripts* de LFS. Al reiniciar su sistema debería poder acceder tanto a Knoppix como a LFS sin problemas. Si es el caso, felicitaciones, ha creado un *Linux From Scratch*.