Linux From Scratch

Versión SVN-20070916

Gerard Beekmans

Linux From Scratch: Versión SVN-20070916

por Gerard Beekmans

Resumen

Traducido por el proyecto *LFS-ES*

Versión de la traducción: 20070916 del 16 de Septiembre de 2007

Tabla de contenidos

Prólogo

Prefacio

Mis aventuras con Linux empezaron en 1998 cuando descargué e instalé mi primera distribución. Tras trabajar cierto tiempo con ella descubrí algunos aspectos que definitivamente quería ver mejorados. Por ejemplo, no me gustaba la forma en la que estaban organizados los guiones de arranque. Intenté con otras distribuciones para solventar estos detalles, pero todas tenían sus ventajas e inconvenientes. Llegué a darme cuenta de que si quería estar completamente satisfecho con el sistema Linux, tenía que construir el mío propio desde cero.

¿Qué significaba esto? Decidí no utilizar paquetes precompilados de ningún tipo, ni CD-ROMs o discos de arranque que instalasen las utilidades básicas. Quería usar mi sistema Linux actual para desarrollar mi propio sistema personalizado. Este sistema Linux "perfecto" debería tener toda la potencia de los otros sistemas sin sus debilidades. Al principio, la idea fue bastante desalentadora, pero me mantuve aferrado a la idea de que podía construir un sistema que tuviese en consideración mis necesidades y deseos en vez de usar un estándar que no se ajustaba a lo que andaba buscando.

Después de sortear todos los problemas de dependencias circulares y errores de compilación, creé un sistema Linux personalizado hecho a medida y completamente funcional. Este proceso me permitió además crear un sistema compacto y ajustado que era más rápido y ocupaba menos espacio que cualquier sistema operativo tradicional. Llamé a este sistema Linux From Scratch (Linux Desde Cero), o sistema LFS para acortar.

Cuando compartí mis metas y experiencias con otros miembros de la comunidad Linux se hizo palpable que había un amplio interés en las ideas que surgieron de mis aventuras con Linux. No sólo porque dicho sistema LFS de construcción personalizada podía cubrir las especificaciones y requerimientos del usuario, sino también porque ofrecía una gran oportunidad para el aprendizaje a los programadores y administradores de sistemas y ampliar su conocimiento sobre Linux. Con este creciente interés nació el Proyecto Linux From Scratch.

El libro *Linux From Scratch* otorga a los lectores el conocimento y las instrucciones para diseñar y construir un sistema Linux a medida. Este libro resalta el proyecto Linux From Scratch y los beneficios que conlleva el uso de este sistema. Los usuarios pueden definir todos los aspectos de su sistema, incluida la jerarquía de directorios, los guiones de arranque y la seguridad. El sistema resultante se compilará por completo a partir del código fuente y el usuario podrá especificar dónde, por qué y cómo se instalarán los programas. Este libro permite a sus lectores adaptar por completo sus sistemas Linux según sus propias necesidades y ofrece a los usuarios un mayor control sobre el sistema.

Espero que paséis buenos momentos trabajando en vuestro sistema LFS y que disfrutéis de los numerosos beneficios de tener un sistema que es realmente *vuestro*.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Audiencia

Existen muchas razones por las que alguien querría leer este libro. La principal razón es instalar un sistema Linux a partir del código fuente. La pregunta que mucha gente se hace es "¿Por qué pasar por todo el embrollo de instalar manualmente un sistema Linux desde cero cuando te puedes limitar a descargar e instalar uno ya existente?". Es una buena pregunta y es el motivo de esta sección del libro.

Una importante razón para la existencia de LFS es enseñar a la gente cómo trabaja internamente un sistema Linux. Construir un sistema LFS ayuda a demostrar lo que hace que Linux funcione, cómo trabajan juntas las distintas partes y cómo unas dependen de otras. Una de las mejores cosas que este proceso de aprendizaje proporciona es la habilidad para adaptar Linux a tus propios gustos y necesidades.

Uno de los beneficios claves de LFS es que tienes el control de tu sistema sin tener que confiar en la implementación de Linux de nadie. Con LFS *tu* estás en el asiento del conductor y puedes dictar cada aspecto de tu sistema, como la estructura de directorios y la configuración de los guiones de arranque. También podrás decidir dónde, por qué y cómo se instalan los programas.

Otro beneficio de LFS es que puedes crear un sistema Linux verdaderamente compacto. Cuando instalas una distribución normal acabas instalando muchos programas que probablemente nunca usarás. Tan sólo están ahí ocupando espacio de disco o peor aún, ciclos de CPU. No es muy difícil conseguir un sistema LFS instalado en menos de 100 MB, lo que es notablemente más pequeño que la mayoría de instalaciones existentes. ¿Todavía te parece demasiado? Algunos de nosotros hemos estado trabajando para crear un sistema LFS embebido realmente pequeño. Hemos instalado un sistema que contiene lo suficiente para ejecutar un servidor web Apache utilizando tan sólo 8 MB de espacio en disco. Con un repaso adicional para reducirlo, se podría llegar a 5 MB o menos. Intenta eso con una distribución normal. Esta es una de las muchas ventajas que te ofrece diseñar tu propio sistema Linux.

Podríamos comparar una distribución de Linux con una hamburguesa que compras en un restaurante de comida rápida. No tienes idea de lo que te estás comiendo. En cambio, LFS no te da una hamburguesa, sino la receta para hacer la hamburguesa. Te permite revisarla, eliminar los ingredientes no deseados y añadir tus propios ingredientes para mejorar el sabor de tu hamburguesa. Cuando estés satisfecho con la receta entonces empiezas a prepararla. Tu la cocinas de la forma que prefieres: asada, cocida, frita o a la barbacoa.

Otra analogía que podemos usar es comparar a LFS con una casa terminada. LFS te dará los planos de la casa, pero tú debes construirla. Tienes libertad para adaptar los planos durante el proceso, para adaptarlos a tus necesidades y preferencias.

Una última ventaja de un sistema Linux hecho a la medida es la seguridad. Compilando el sistema entero a partir del código fuente tienes la posibilidad de supervisar todo y aplicar todos los parches de seguridad que creas que son necesarios. No tienes que esperar a que alguien te proporcione un nuevo paquete binario que corrija un problema de seguridad. A no ser que examines el nuevo parche y lo implantes por ti mismo no tienes garantía de que ese nuevo paquete se haya construido correctamente y realmente solucione el problema.

El obejtivo de LFS es construir un sistema basado en niveles completo y utilizable. Los lectores que no deseen construir su propio sistema LFS no se podrán beneficiar de la información que hay en este libro. Si sólo quieres saber lo que sucede mientras arranca tu ordenador, entonces te recomendamos el “From Power Up To Bash Prompt” HOWTO (De La Puesta En Marcha Al Indicador De Bash CÓMO) que podrás encontrar en *http://axiom.anu.edu.au/~okeefe/p2b/* o en el sitio web The Linux Documentation Project (TLDP) *http://www.tldp.org/HOWTO/From-PowerUp-To-Bash-Prompt-HOWTO.html*. Este CÓMO construye un sistema que es similar al de este libro, pero lo enfoca estrictamente hacia la creación de un sistema capaz de iniciar el símbolo del sistema de BASH. Considera tu objetivo. Si lo que quieres es construirte tu propio sistema Linux y aprender mientras lo haces, este libro es la mejor opción.

Hay muy buenas razones para construir tu propio sistema LFS aparte de las aquí listadas. Esta sección es sólo la punta del iceberg. A medida que avances en tu experiencia con LFS encontrarás por ti mismo el poder que la información y el conocimiento realmente brindan.

Prerrequisitos

Construir un sistema LFS no es una tarea fácil. Se necesita tener un cierto nivel de conocimientos en la administración de sistemas Unix para poder resolver problemas y ejecutar correctamente los comandos listados. En particular, y como mínimo imprescindible, el lector debería tener la habilidad para usar la línea de comandos (shell) para copiar o mover ficheros y directorios, listar directorios y el contenido de ficheros, y cambiar de directorio. También se espera que el lector tenga un conocimiento razonable sobre el uso y la instalación de software Linux.

Debido a que el libro asume *al menos* este nivel basico, es improblable que los diversos foros de soporte de LFS puedan proporcionarte mucha ayuda al respecto. Encontrarás que tus preguntas sobre dichos conocimientos básicos no serán respondidas, o simplemente serás reenviado a la lista de lecturas prévias esenciales del LFS.

Antes de construir un sistema LFS, recomendamos que leas los siguientes CÓMOS:

- Software-Building-HOWTO (Construcción de Software CÓMO):
  http://www.tldp.org/HOWTO/Software-Building-HOWTO.html

  Esta es una guía asequible sobre cómo construir e instalar los paquetes de software Unix “genéricos” bajo Linux.
- The Linux Users' Guide (La Guía del Usuario de Linux).
  Versión en castellano:
  http://es.tldp.org/Manuales-LuCAS/GLUP/glup_0.6-1.1-html-1.1
  Versión en inglés:
  http://www.linuxhq.com/guides/LUG/guide.html

  Esta guía cubre el uso de una amplia gama de software Linux.
- The Essential Pre-Reading Hint (Receta de las lecturas previas esenciales):
  http://www.linuxfromscratch.org/hints/downloads/files/essential_prereading.txt

  Esta es una receta del LFS escrita específicamente para los nuevos usuarios de Linux. Incluye un listado de enlaces a excelentes fuentes de información sobre un amplio rango de tópicos. Cualquier persona que intente instalar LFS debería comprender muchos de los tópicos mencionados en esta receta.

Requisitos del sistema anfitrión

Tu sistema anfitrión debería tener el siguiente software con las versiones mínimas indicadas. Esto no debería ser un problema para la mayoría de las distribuciones Linux modernas. Debes tener en cuenta también que muchas distribuciones ponen las cabeceras del software en paquetes separados, con frecuencia de la forma “<nombre-del-paquete>-devel” o “<nombre-del-paquete>-dev”. Asegurate de instalarlos si tu distribución los proporciona.

- **Bash-2.05a**
- **Binutils-2.12** (No se recomiendan las versiones superiores a 2.18 debido a que no han sido probadas)
- **Bison-1.875**
- **Bzip2-1.0.2**
- **Coreutils-5.0** (o Sh-Utils-2.0, Textutils-2.0 y Fileutils-4.1)
- **Diffutils-2.8**
- **Findutils-4.1.20**
- **Gawk-3.0**
- **Gcc-3.0.1** (No se recomiendan las versiones superiores a 4.2.1 debido a que no han sido probadas)
- **Glibc-2.2.5** (No se recomiendan las versiones superiores a 2.6.1 debido a que no han sido probadas)
- **Grep-2.5**

- **Gzip-1.2.4**
- **Linux Kernel-2.6.x** (compilado con GCC-3.0 o superior)

  La razón por la que se requiere tal versión del núcleo es que el soporte para almacenaniento local de hilos de Binutils no será compilado y el banco de pruebas de NPTL (Native POSIX Threading Library) fallará si el núcleo del anfitrión no es al menos una versión 2.6.x compilada con una versión de GCC 3.0 o superior.

  Si el núcleo del anfitrión no es 2.6.x y no ha sido compilado usando GCC-3.0 (o superior), tendrás que reemplazar el núcleo con uno que cumpla las especificaciones. Tienes dos métodos para hacer esto. Primero, mira si tu distribuidor Linux proporciona un paquete con el núcleo 2.6. Si es así, puede que desees instalarlo. Si tu distribuidor no ofrece un paquete del núcleo 2.6 o prefieres no instalarlo, entonces puedes compilar tu mismo un núcleo 2.6. Las instrucciones para compilar el núcleo y configurar el gestor de arranque (suponiendo que el anfitrión utilice GRUB) se encuentran en Capítulo 8.
- **Make-3.79.1**
- **Patch-2.5.4**
- **Sed-3.0.2**
- **Tar-1.14**
- **Texinfo-4.8**

Para saber si en tu sistema anfitrión se encuentran todas las versiones correctas, ejecuta lo siguiente:

```
cat > version-check.sh << "EOF"
#!/bin/bash

# Sencillo guión para listar los números de versión de herramientas
# de desarrollo críticas

bash --version | head -n1 | cut -d" " -f2-4
echo -n "Binutils: "; ld --version | head -n1 | cut -d" " -f3-4
bison --version | head -n1
bzip2 --version 2>&1 < /dev/null | head -n1 | cut -d" " -f1,6-
echo -n "Coreutils: "; chown --version | head -n1 | cut -d")" -f2
diff --version | head -n1
find --version | head -n1
gawk --version | head -n1
gcc --version | head -n1
/lib/libc.so.6 | head -n1 | cut -d" " -f1-7
grep --version | head -n1
gzip --version | head -n1
cat /proc/version
make --version | head -n1
patch --version | head -n1
sed --version | head -n1
tar --version | head -n1
makeinfo --version | head -n1

EOF

bash version-check.sh
```

Tipografía

A lo largo del libro se utilizan ciertas convenciones tipogáfricas con el objeto de facilitar la comprensión. Esta sección contiene algunos ejemplos del formato tipográfico que encontrarás en Linux From Scratch:

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

Este tipo de texto está diseñado para teclearse exactamente como aparece, a menos que se indique lo contrario en el texto subyacente. También se utiliza en las secciones explicativas para identificar el comando al que se hace referencia.

```
install-info: unknown option '--dir-file=/mnt/lfs/usr/info/dir'
```

Este tipo de texto (texto de ancho fijo) representa salida por pantalla, probablemente como resultado de la ejecución de comandos. También se usa para especificar nombres de ficheros, como `/etc/ld.so.conf`.

Enfasis

Este tipo de texto se utiliza con varios fines en el libro. Su objetivo principal es poner de relieve puntos importantes.

http://www.linuxfromscratch.org/

Este tipo de texto se usa para hipervínculos, tanto dentro de la comunidad LFS como a páginas exteriores. Esto incluye direcciones de descarga, CÓMOs o sitios web.

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

Este formato se usa para la creación de ficheros de configuración. El primer comando solicita al sistema que cree el fichero `$LFS/etc/group` a partir de lo que se teclee en las líneas siguientes, hasta encontrar la secuencia de fín de fichero (EOF). Por lo tanto, la sección entera debe teclearse tal cual.

<TEXTO A REEMPLAZAR>

Este formato se utiliza para encapsular texto que no debe ser escrito tal y como aparece.

[TEXTO OPCIONAL]

Este formato se utiliza para encapsular texto que es opcional.

`passwd(5)`

Este formato se usa para referirse a una página de manual específica. El número entre paréntesis indica la sección concreta dentro de **man**. Por ejemplo, **passwd** tiene dos páginas de manual. Siguiendo las instrucciones de instalación del LFS, dichas páginas se encontrarán en `/usr/share/man/man1/passwd.1` y `/usr/share/man/man5/passwd.5`. Ambas contienen diferente información. Cuando el libro utiliza `passwd(5)` se refiere exactamente a `/usr/share/man/man5/passwd.5`. **man passwd** mostrará la primera página de manual que encuentre referente a “passwd”, que será `/usr/share/man/man1/passwd.1`. Para este ejemplo, tendrás que ejecutar **man 5 passwd** para leer la página de manual específica que se referencia. Debería tenerse en cuenta que muchas páginas de manual no tienen nombres duplicados en diferentes secciones. Por tanto, **man *<nombre del programa>*** suele ser suficiente.

Estructura

Este libro se divide en las siguientes partes:

Parte I - Introducción

En la Parte I se explican algunas cosas importantes sobre cómo hacer la instalación de LFS. También ofrece información general sobre el libro.

Parte II - Preparativos para la construcción

La Parte II describe cómo preparar el proceso de construcción: crear una partición, descargar los paquetes y compilar las herramientas temporales.

Parte III - Construcción del sistema LFS

La Parte III te guía a través de la construcción del sistema LFS: compilar e instalar todos los paquetes uno por uno, activar los guiones de arranque e instalar el núcleo. El sistema Linux obtenido es la base sobre la que podrás construir más software, ampliando tu sistema del modo que prefieras. Al final del libro encontrarás un listado de todos los programas, librerías y ficheros importantes que se han instalado, a modo de referencia rápida.

Errata

El software usado para crear un sistema LFS se actualiza y mejora constantemente. Avisos de seguridad y correcciones de errores pueden estar disponibles despues de publicar el libro LFS. Para comprobar si las versiones de los paquetes o las instrucciones de este versión del LFS necesitan cualquier modificación para solvertar problemas de seguridad o corregir otros errores, visita *http://www.linuxfromscratch.org/lfs/errata/development/* antes de comenzar la construcción. Deberías tener en cuenta cualquier cambio mencionado y aplicarlo en la sección apropiada del libro a medida que avances en la construcción del sistema LFS.

Parte I. Introducción

Capítulo 1. Introducción

1.1. Cómo construir un sistema LFS

El sistema LFS se construirá utilizando una distribución Linux ya instalada (como Debian, Mandriva, RedHat o SUSE). Este sistema Linux existente (el anfitrión) se utilizará como punto de inicio para suministrar los programas necesarios, como un compilador, un enlazador y un intérprete de comandos, para construir el nuevo sistema. Selecciona la opción “desarrollo” durante la instalación de la distribución para poder acceder a estas herramientas.

Como alternativa a la instalación previa de otra distribución, puede que prefieras utilizar el LiveCD de Linux From Scratch. El CD funciona bien como sistema anfitrión, proporcionando todas las herramientas que necesitarás para seguir con éxito las instrucciones de este libro. Una vez que tengas el CD ya no es necesario tener conexión de red o hacer descargas adicionales. Para más información sobre el LiveCD de LFS o descargar una copia, visita *http://www.linuxfromscratch.org/livecd/*

Nota

El LiveCD de LFS puede no funcionar en configuraciones de hardware recientes, fallando en el arranque o fallando al detectar algunos dispositivos, como los discos duros SATA.

El equipo del LiveCD de LFS está trabajando pora solucionar dichos problemas, pero necesitan tu ayuda testeandolo, avisando de los problemas encontrados y colaborando en el desarrollo del LiveCD.

Por favor, envía tus informes sobre el LiveCD de LFS o las colaboraciones para ayudar en su desarrollo a la *lista de correo del Live CD de LFS*.

El Capítulo 2 de este libro describe cómo crear una nueva partición nativa Linux y un sistema de ficheros, el sitio donde se compilará e instalará el nuevo sistema LFS. El Capítulo 3 explica qué paquetes y parches deben descargarse para construir un sistema LFS y cómo guardarlos en el nuevo sistema de ficheros. El Capítulo 4 muestra cómo configurar un entorno de trabajo adecuado. Por favor, lee con detenimiento el Capítulo 4, pues explica diversos temas importantes a tener en cuenta antes de empezar a trabajar en el Capítulo 5 y posteriores.

En el Capítulo 5 se describe la instalación de una serie de paquetes que formarán el entorno básico de desarrollo (o herramientas principales) utilizado para construir el sistema real en el Capítulo 6. Varios de estos paquetes son necesarios para resolver dependencias circulares. Por ejemplo, para compilar un compilador necesitas un compilador.

El Capítulo 5 muestra también al usuario cómo construir en una primera fase las herramientas principales, compuestas por Binutils y GCC (primera fase significa, básicamente, que estos dos paquetes centrales serán reinstalados más tarde). El siguiente paso es construir Glibc, la librería C. Glibc será compilada con los programas de las herramientas principales construidas en la primera fase. Entonces se construirá una segunda fase de las herramientas principales. Esta vez se enlazarán dinámicamente contra la recién construida Glibc. Todos los restantes paquetes del Capítulo 5 se construirán usando esta segunda fase de las herramientas principales. Cuando esto esté hecho, el proceso de instalación de LFS ya no dependerá de la distribución anfitriona, con la excepción del núcleo en ejecución.

Este esfuerzo para aislar el nuevo sistema de la distribución anfitriona puede parecer excesivo, pero en Sección 5.2, “Notas técnicas sobre las herramientas” se da una explicación técnica completa.

En el Capítulo 6 se construye el auténtico sistema LFS. Se utiliza el programa **chroot** (change root, cambio de raíz) para entrar en un entorno virtual y ejecutar un nuevo intérprete de comandos cuyo directorio raíz será la partición LFS. Esto es muy similar a reiniciar e indicarle al núcleo que monte la partición LFS como partición raíz. El sistema no es realmente reiniciado, si no que se cambia la raíz, porque crear un sistema arrancable requiere un trabajo adicional que

no es necesario aún. La mayor ventaja es que “cambiar la raíz” permite seguir usando el sistema anfitrión mientras se construye el LFS. Mientras espera que se complete la compilación de un paquete, el usuario puede cambiar a otra consola virtual (VC) o escritorio X y continuar usando el ordenador normalmente.

Para terminar la instalación, en el Capítulo 7 se configuran los guiones de arranque, y el núcleo y el gestor de arranque se configuran en el Capítulo 8. El Capítulo 9 contiene información para profundizar en la experiencia LFS después de este libro. Tras completar los pasos de este libro, el ordenador estará preparado para reiniciarse dentro del nuevo sistema LFS.

Este es el proceso en pocas palabras. La información detallada sobre cada paso a dar se expone en los siguientes capítulos y descripciones de los paquetes. Los temas que pueden parecer complicados se aclararán y todo estará en su sitio a medida que te embarques en la aventura del LFS.

1.2. Qué hay de nuevo desde la última publicación

A continución hay una lista de los paquetes actualizados desde la anterior publicación del libro.

Actualizado a:

- Berkeley DB 4.6.19
- Binutils 2.18
- GCC 4.2.1
- Glibc 2.6.1
- LFS-Bootscripts 20070813
- Linux 2.6.22.6
- Man-pages 2.64
- Sysklogd 1.5
- udev-config-20070731

Añadido:

- bash-3.2-fixes-6.patch
- readline-5.2-fixes-4.patch
- vim-7.1-fixes-2.patch

Eliminado:

- bash-3.2-fixes-5.patch
- db-4.5.29-fixes-1.patch
- gcc-4.1.2-specs-1.patch
- readline-5.4-fixes-4.patch
- sysklogd-1.4.1-8bit-1.patch
- sysklogd-1.4.1-fixes-2.patch
- vim-7.1-fixes-1.patch

1.3. Historial de modificaciones

Esta es la versión 20070916 del día 16 de Septiembre de 2007 de la traducción al castellano de la versión SVN-20070916 del libro Linux From Scratch publicado el 16 de Septiembre de 2007. Si este libro tiene más de seis meses de antigüedad es probable que haya disponible una versión más nueva y mejor. Para encontrarlo, comprueba uno de los servidores alternativos listados en *http://www.linuxfromscratch.org/mirrors.html*.

A continuación hay una lista con los cambios realizados desde la anterior versión del libro.

Cambios:

- 2007-09-16
  - [manuel] - Actualizada la lista de contenidos de Ncurses y corregidos varios errores textuales. Gracias a Chris Staub por el parche.
- 2007-09-15
  - [matthew] - Añadidos los últimos parches del desarrollador para Vim.
  - [matthew] - Actualizado a Sysklogd-1.5. Corrige *#2055*.
  - [matthew] - Añadidos los últimos parches del desarrollador para Readline. Corrige *#2068*.
  - [matthew] - Actualizado a Man-pages 2.64. Corrige *#2061*.
  - [matthew] - Actualizado a Linux-2.6.22.6. Corrige *#2070*.
  - [jhuntwork] - Actualizado a Glibc-2.6.1. Corrige *#2018*. Gracias a Matthew Burgess por preparar el parche, Robert Connolly y Dan Nicholson por investigar la mejor forma de ajustar CFLAGS, y Greg Schafer por mostrar los beneficios técnicos del uso de CFLAGS con Glibc.
  - [jhuntwork] - Actualizado a GCC-4.2.1. Corrige *#2002*. Gracias a Matthew Burgess por preparar el parche.
  - [matthew] - Actualizado a DB-4.6.19. Corrige *#2051*.
  - [matthew] - Actualizado a Binutils-2.18. Corrige *#2069*.
  - [matthew] - Añadidos los últimos parches del desarrollador para Bash. Corrige *#2067*.
- 2007-09-07
  - [manuel] - Añadidos bloques de matainformación sect1info en las páginas de paquetes para facilitar el soporte de administradores de paquetes en jhalfs.

Publicado el LFS 6.3 el 28 de Agosto de 2007.

1.4. Recursos

1.4.1. FAQ

Si durante la construcción del sistema LFS encuentras algún fallo, tienes preguntas, o encuentras un error tipográfico en el libro, consulta primero las FAQ (Preguntas Hechas Frecuentemente) que se encuentran en *http://www.linuxfromscratch.org/faq/*.

1.4.2. Listas de correo

El servidor `linuxfromscratch.org` hospeda una serie de listas de correo utilizadas para el desarrollo del proyecto LFS. Estas incluyen, entre otras, las listas principales de desarrollo y soporte. Si la FAQ no resuelve tus problemas, el siguiente paso debería ser buscar en las listas de correo en *http://www.linuxfromscratch.org/search.html*.

Para obtener información relacionada con las listas disponibles, cómo suscribirse a ellas, localización de los archivos, etc, visita *http://www.linuxfromscratch.org/mail.html*.

La comunidad hispanoparlante dispone de dos listas de correo que no pertenecen al servidor `linuxfromscratch.org`:

- Soporte, ayuda y noticias sobre LFS:

https://www.champinet.com/cgi-bin/mailman/listinfo/linux-desde-cero

- Coordinación de la traducción de LFS al castellano:
  http://listas.escomposlinux.org/mailman/listinfo/lfs-es

1.4.3. IRC

Varios miembros de la comunidad LFS ofrecen asistencia técnica en nuestro servidor IRC. Antes de utilizar este método de ayuda te pedimos que compruebes si en las FAQ de LFS o en los archivos de las listas de correo se encuentra la respuesta a tu problema. Puedes entrar al servidor IRC a través de `irc.linuxfromscratch.org`. El canal de soporte se llama #LFS-support.

1.4.4. Referencias

En la página "LFS Package Reference", en *http://www.linuxfromscratch.org/~matthew/LFS-references.html*, tienes a tu disposición unos apuntes útiles con información adicional sobre los paquetes.

1.4.5. Servidores alternativos

El proyecto LFS tiene por todo el mundo varios servidores alternativos para facilitar el acceso a las páginas web y la descarga de los paquetes requeridos. Por favor, visita el sitio web *http://www.linuxfromscratch.org/mirrors.html* para consultar la lista de los servidores alternativos actuales.

El proyecto LFS-ES, que se ocupa de la traducción al castellano de los textos del LFS, dispone de los siguientes servidores:

- EcolNet, España [Varios servidores ADSL] - *http://www.escomposlinux.org/lfs-es/*
- Dattatec.com, Argentina [100 Mbits] - *http://www.lfs-es.info/*
- Balaguer, España [ADSL 512 Kbits de salida] - *http://www.macana-es.com/*

1.4.6. Información de contacto

Por favor, envía todas tus preguntas y comentarios a una de las listas de correo de LFS o LFS-ES (ver arriba).

1.5. Ayuda

Si mientras estás usando este libro te surge algún problema o duda, consulta primero las FAQ que hay en *http://www.linuxfromscratch.org/faq/#generalfaq*. Probablemente tu pregunta esté contestada aquí. Si no es así, prueba a encontrar la fuente del problema. La siguiente receta puede darte algunas ideas para encontrar la solución: *http://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.

Si no puedes encontrar tu problema en la FAQ, busca en las listas de correo en *http://www.linuxfromscratch.org/search.html*.

También tenemos una maravillosa comunidad LFS que está encantada de ofrecer ayuda a través las listas de correo y del canal IRC (mira el Capítulo 1 - Listas de correo). Sin embargo, cada día recibimos cantidad de peticiones de ayuda, y muchas de ellas pueden ser fácilmente resueltas consultando primero la FAQ o buscando en las listas de correo. Así que para ofrecerte la mejor asistencia posible, primero necesitas hacer cierta investigación por tu cuenta. Esto nos permite centrarnos en las cuestiones de soporte menos habituales. Si en tu búsqueda no encuentras la solución, por favor, incluye toda la información necesaria (mencionada a continuación) en tu petición de ayuda.

1.5.1. Cosas a mencionar

Además de una breve explicación del problema experimentado, las cosas esenciales que se deben incluir en la petición de ayuda son:

- La versión del libro que se está usando (en este caso, SVN-20070916).
- La distribución anfitriona (y su versión) usada como base para crear el LFS.
- El paquete o sección en el que se encontró el problema.
- El mensaje de error exacto o los síntomas que aparecen.
- Si te has desviado o no del libro.

Nota

Desviarse del libro *no* implica que no vayamos a ayudarte. Después de todo, LFS se basa en la elección. Avisarnos sobre cualquier cambio en el procedimiento establecido nos ayudará a detectar las posibles causas de tu problema.

1.5.2. Problemas con el guión configure

Cuando algo va mal mientras se ejecuta el guión **configure**, consulta el fichero `config.log`. Este fichero puede contener errores encontrados por **configure** que no se mostraron en pantalla. Incluye las líneas *relevantes* si necesitas pedir ayuda.

1.5.3. Problemas de compilación

Tanto la salida por pantalla como el contenido de varios ficheros son útiles para determinar la causa de los problemas de compilación. La salida por pantalla del guión **configure** y del comando **make** pueden ser útiles. No es necesario incluir toda la salida, pero incluye suficiente información relevante. A continuación hay un ejemplo del tipo de información a incluir de una salída por pantalla de **make**:

```
gcc -DALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-DLOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-DLIBDIR=\"/mnt/lfs/usr/lib\"
-DINCLUDEDIR=\"/mnt/lfs/usr/include\" -DHAVE_CONFIG_H -I. -I.
-g -O2 -c getopt1.c
gcc -g -O2 -static -o make ar.o arscan.o commands.o dir.o
expand.o file.o function.o getopt.o implicit.o job.o main.o
misc.o read.o remake.o rule.o signame.o variable.o vpath.o
default.o remote-stub.o version.o opt1.o
-lutil job.o: In function `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: undefined reference
to `getloadavg'
collect2: ld returned 1 exit status
make[2]: *** [make] Error 1
make[2]: Leaving directory `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

En este caso, mucha gente simplemente incluye la sección final a partir de:

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

Esto no es suficiente información para diagnosticar el problema porque sólo nos dice que algo fue mal, no *qué* fue mal. Lo que se debería incluir para resultar útil es la sección completa tal y como aparece en el ejemplo anterior, ya que incluye el comando que se estaba ejecutando y sus mensajes de error.

En *http://catb.org/~esr/faqs/smart-questions.html* hay disponible un artículo excelente sobre cómo buscar ayuda en Internet. Lee y sigue los consejos de este documento para aumentar las posibilidades de obtener la ayuda que necesitas.

Parte II. Preparativos para la construcción

Capítulo 2. Preparar una nueva partición

2.1. Introducción

En este capítulo se preparará la partición que contendrá el sistema LFS. Se creará la propia partición, se creará un sistema de ficheros en ella y se montará.

2.2. Crear una nueva partición

Como muchos otros sistemas operativos, LFS se instala normalmente en una partición dedicada. El método recomendado para construir un sistema LFS es utilizar una partición libre vacía o, si tienes suficiente espacio sin particionar, crear una. Sin embargo, un sistema LFS (de hecho incluso varios sistemas LFS) puede instalarse también en una partición que ya esté ocupada por otro sistema operativo, y los diferentes sistemas coexistirán pacíficamente. El documento *http://www.linuxfromscratch.org/hints/downloads/files/lfs_next_to_existing_systems.txt* explica cómo implementar esto, mientras que este libro muestra el método para utilizar una nueva partición en la instalación.

Un sistema mínimo necesita una partición de 1,3 GB más o menos. Esto es suficiente para almacenar todos los archivos de código fuente y compilar los paquetes. Sin embargo, si se piensa usar el sistema LFS como sistema Linux principal probablemente se instalará software adicional, necesitando más espacio (2-3 GB). El propio sistema LFS no ocupa mucho espacio. Una gran parte de este espacio es requerido para proporcionar suficiente espacio libre temporal. Compilar paquetes puede necesitar mucho espacio en disco que será liberado tras instalar el paquete.

Como casi nunca hay suficiente memoria RAM disponible para los procesos de compilación, es buena idea utilizar una pequeña partición como espacio de intercambio (`swap`). Este espacio lo usa el núcleo para almacenar los datos menos usados y hacer sitio en memoria para los procesos activos. La partición de intercambio para el sistema LFS puede ser la misma del sistema anfitrión, por lo que no hace falta crear otra si el sistema anfitrión tiene una activada.

Inicia un programa de particionado como **cfdisk** o **fdisk** pasándole como argumento el nombre del disco duro en el que debe crearse la nueva partición, por ejemplo `/dev/hda` para el disco IDE primario. Crea una partición Linux nativa y, si hace falta, una partición de intercambio. Por favor, consulta `cfdisk(8)` o `fdisk(8)` si todavía no sabes cómo usar estos programas.

Recuerda la denominación de tu nueva partición (por ejemplo, `hda5`). Este libro se referirá a ella como la partición LFS. Recuerda también la denominación de la partición de intercambio. Estos nombres se necesitarán posteriormente para el fichero `/etc/fstab`.

2.3. Crear un sistema de ficheros en la partición

Ahora que hay preparada una partición en blanco ya puede crearse el sistema de ficheros. El más usado en el mundo de Linux es el llamado “second extended file system” (segundo sistema de ficheros extendido, o `ext2`), pero con la gran capacidad de los discos duros actuales los llamados sistemas de ficheros con registro de transacciones (journaling) se han hecho muy populares. El tercer sistema de ficheros extendido (`ext3`) es una evolución ámpliamente usada de `ext2`, que añade capacidad de registro de transacciones y es compatible con las utilidades de E2fsprogs. Crearemos un saitema de ficheros `ext3`. En *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/filesystems.html* podrás encontrar la instrucciones para construir otros sistemas de ficheros.

Para crear un sistema de ficheros `ext3` en la partición LFS, ejecuta lo siguiente:

```
mke2fs -jv /dev/<xxx>
```

Sustituye <*xxx*> por el nombre de la partición LFS (hda5 en nuestro ejemplo anterior).

Nota

Algunas distribuciones usadas como anfitrión utilizan características personalizadas en sus herramientas de creación de sistemas de ficheros (E2fsprogs). Esto puede causar problemas cuando arranques tu nuevo LFS en el Capítulo 9, pues dichas características no estarán soportadas por el E2fsprogs instalado en LFS. Obtendrás un error similar a "unsupported filesystem features, upgrade your e2fsprogs". Para comprobar si tu sistema anfitrión utliza ampliaciones personalizadas, ejecuta el siguiente comando:

```
debugfs -R feature /dev/<xxx>
```

Si la salida contiene características diferentes a: has_journal, dir_index, filetype, large_file, resize_inode, sparse_super or needs_recovery, entonces tu sistema anfitrión posiblemente tenga ampliaciones personalizadas. En este caso, para evitar posteriores problemas, deberías compilar el paquete e2fsprogs base y utilizar los binarios resultantes para recrear el sistema de ficheros de tu partición LFS:

```
cd /tmp
tar -xjvf /ruta/a/sources/e2fsprogs-1.40.2.tar.bz2
cd e2fsprogs-1.40.2
mkdir -v build
cd build
../configure
make #advierte que no se hace 'make install' aquí!
./misc/mke2fs -jv /dev/<xxx>
cd /tmp
rm -rfv e2fsprogs-1.40.2
```

Si usas una partición de intercambio (swap), no es necesario formatearla. Si creas una neva partición de intercambio, deberás inicializarla ejecutando el siguiente comando:

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

Sustituye <*yyy*> por el nombre de la partición de intercambio.

2.4. Montar la nueva partición

Ahora que se ha creado un sistema de ficheros es necesario hacer accesible la partición. Para esto debe montarse en el punto de montaje elegido. Para los propósitos de este libro se asume que el sistema de ficheros se monta en /mnt/lfs, pero la elección del directorio se deja para tí.

Elige un punto de montaje y asígnalo a la variable de entorno LFS ejecutando:

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

Crea el punto de montaje y monta el sistema de ficheros LFS ejecutando:

```
mkdir -pv $LFS
mount -v -t ext3  /dev/<xxx> $LFS
```

Sustituye <*xxx*> por el nombre de la partición LFS.

Si utilizas múltiples particiones para LFS (digamos que una para `/` y otra para `/usr`) móntalas usando:

```
mkdir -pv $LFS
mount -v -t ext3  /dev/<xxx> $LFS
mkdir -v $LFS/usr
mount -v -t ext3 /dev/<yyy> $LFS/usr
```

Sustituye *<xxx>* e *<yyy>* por los nombres de partición apropiados.

Asegúrate de que esta nueva partición no se monte con permisos muy restrictivos (como las opciones `nosuid`, `nodev` o `noatime`). Ejecuta el comando **mount** sin parámetros para ver con qué opciones está montada la partición LFS. Si ves `nosuid`, `nodev` o `noatime`, necesitarás remontarla.

Si estás usando una partición `swap`, asegurate de que está activada usando el comando **swapon**:

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

Reenplaza *<zzz>* con el nombre de la partición de intercambio.

Ahora que se ha establecido un lugar en el que trabajar, es hora de descargar los paquetes.

Capítulo 3. Paquetes y parches

3.1. Introducción

Este capítulo incluye una lista con los paquetes que se han de descargar para construir un sistema Linux básico. Los números de versión listados corresponden a versiones de los programas que se sabe que funcionan y este libro se basa en ellos. Recomendamos encarecidamente que no uses versiones más nuevas, pues los comandos de construcción para una versión puede que no funcionen con la nueva. Los paquetes más nuevos pueden también tener problemas que necesiten soluciones. Dichas soluciones se desarrollarán y estabilizarán en la versión de desarrollo del libro.

Las localizaciones de descarga puede que no estén siempre disponibles. En el caso de que una localización de descarga haya cambiado desde la publicación de este libro, Google (*http://www.google.com/*) es una útil herramienta de búsqueda para muchos paquetes. Si la búsqueda no da resultados, prueba uno de los métodos alternativos de descarga listados en *http://www.linuxfromscratch.org/lfs/packages.html*.

Será necesario guardar todos los paquetes y parches descargados en algún sitio que esté disponible durante toda la construcción. También se necesita un directorio de trabajo en el que desempaquetar las fuentes y construirlas. Puede usarse `$LFS/sources` tanto para almacenar los paquetes y parches como directorio de trabajo. Al usar este directorio, los elementos requeridos se encontrarán en la partición LFS y estarán disponibles durante todas las fases del proceso de construcción.

Para crear este directorio, ejecuta el siguiente comando como usuario `root` antes de comenzar la sesión de descarga:

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

Haz este directorio escribible y pegajoso (sticky). “Pegajoso” significa que aunque diversos usuarios tengan permisos de escritura en un mismo directorio, sólo el propietario de un fichero puede borrarlo. El siguiente comando activará los modos de escritura y pegajoso:

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

3.2. Todos los paquetes

Descarga u obtén por otros métodos los siguientes paquetes:

• Autoconf (2.61) - 1,018 KB:
Página web: *http://www.gnu.org/software/autoconf/*
Descarga: *http://ftp.gnu.org/gnu/autoconf/autoconf-2.61.tar.bz2*
Súma MD5: `36d3fe706ad0950f1be10c46a429efe0`

• Automake (1.10) - 873 KB:
Página web: *http://www.gnu.org/software/automake/*
Descarga: *http://ftp.gnu.org/gnu/automake/automake-1.10.tar.bz2*
Súma MD5: `0e2e0f757f9e1e89b66033905860fded`

• Bash (3.2) - 2,471 KB:
Página web: *http://www.gnu.org/software/bash/*
Descarga: *http://ftp.gnu.org/gnu/bash/bash-3.2.tar.gz*
Súma MD5: `00bfa16d58e034e3c2aa27f390390d30`

• Bash Documentation (3.2) - 2,143 KB:
Descarga: *http://ftp.gnu.org/gnu/bash/bash-doc-3.2.tar.gz*
Súma MD5: `0e904cb46ca873fcfa65df19b024bec9`

• Berkeley DB (4.6.19) - 11,600 KB:
Página web: *http://www.oracle.com/technology/software/products/berkeley-db/index.html*
Descarga: *http://download-east.oracle.com/berkeley-db/db-4.6.19.tar.gz*
Súma MD5: `89c7390ff120d5ebf3eccc5f97249e79`

• Binutils (2.18) - 14,612 KB:
Página web: *http://sources.redhat.com/binutils/*
Descarga: *http://ftp.gnu.org/gnu/binutils/binutils-2.18.tar.bz2*
Súma MD5: `9d22ee4dafa3a194457caf4706f9cf01`

• Bison (2.3) - 1,055 KB:
Página web: *http://www.gnu.org/software/bison/*
Descarga: *http://ftp.gnu.org/gnu/bison/bison-2.3.tar.bz2*
Súma MD5: `c18640c6ec31a169d351e3117ecce3ec`

• Bzip2 (1.0.4) - 822 KB:
Página web: *http://www.bzip.org/*
Descarga: *http://www.bzip.org/1.0.4/bzip2-1.0.4.tar.gz*
Súma MD5: `fc310b254f6ba5fbb5da018f04533688`

• Coreutils (6.9) - 5,258 KB:
Página web: *http://www.gnu.org/software/coreutils/*
Descarga: *http://ftp.gnu.org/gnu/coreutils/coreutils-6.9.tar.bz2*
Súma MD5: `c9607d8495f16e98906e7ed2d9751a06`

• DejaGNU (1.4.4) - 1,056 KB:
Página web: *http://www.gnu.org/software/dejagnu/*
Descarga: *http://ftp.gnu.org/gnu/dejagnu/dejagnu-1.4.4.tar.gz*
Súma MD5: `053f18fd5d00873de365413cab17a666`

• Diffutils (2.8.1) - 762 KB:
Página web: *http://www.gnu.org/software/diffutils/*
Descarga: *http://ftp.gnu.org/gnu/diffutils/diffutils-2.8.1.tar.gz*
Súma MD5: `71f9c5ae19b60608f6c7f162da86a428`

• E2fsprogs (1.40.2) - 3,873 KB:
Página web: *http://e2fsprogs.sourceforge.net/*
Descarga: *http://prdownloads.sourceforge.net/e2fsprogs/e2fsprogs-1.40.2.tar.gz*
Súma MD5: `130ce559a0f311ea2bc04a47b4982d0a`

• Expect (5.43.0) - 514 KB:
Página web: *http://expect.nist.gov/*
Descarga: *http://expect.nist.gov/src/expect-5.43.0.tar.gz*
Súma MD5: `43e1dc0e0bc9492cf2e1a6f59f276bc3`

• File (4.21) - 538 KB:
Descarga: *ftp://ftp.gw.com/mirrors/pub/unix/file/file-4.21.tar.gz*
Súma MD5: `9e3503116f4269a1be70220ee2234b0e`

Nota

File (4.21) puede que no esté disponible en la localización indicada. En ocasiones los administradores de la localización principal de descarga eliminan las versiones antiguas cuando se libera una nueva. Puedes encontrar una localización alternativa de descarga con la versión correcta en *http://www.linuxfromscratch.org/lfs/download.html#ftp*.

• Findutils (4.2.31) - 1,296 KB:
Página web: *http://www.gnu.org/software/findutils/*
Descarga: *http://ftp.gnu.org/gnu/findutils/findutils-4.2.31.tar.gz*
Súma MD5: `a0e31a0f18a49709bf5a449867c8049a`

• Flex (2.5.33) - 680 KB:
Página web: *http://flex.sourceforge.net*
Descarga: *http://prdownloads.sourceforge.net/flex/flex-2.5.33.tar.bz2*
Súma MD5: `343374a00b38d9e39d1158b71af37150`

• Gawk (3.1.5) - 1,716 KB:
Página web: *http://www.gnu.org/software/gawk/*
Descarga: *http://ftp.gnu.org/gnu/gawk/gawk-3.1.5.tar.bz2*
Súma MD5: `5703f72d0eea1d463f735aad8222655f`

• GCC (4.2.1) - 43,029 KB:
Página web: *http://gcc.gnu.org/*
Descarga: *http://ftp.gnu.org/gnu/gcc/gcc-4.2.1/gcc-4.2.1.tar.bz2*
Súma MD5: `cba410e6ff70f7d7f4be7a0267707fd0`

• Gettext (0.16.1) - 8,340 KB:
Página web: *http://www.gnu.org/software/gettext/*
Descarga: *http://ftp.gnu.org/gnu/gettext/gettext-0.16.1.tar.gz*
Súma MD5: `3d9ad24301c6d6b17ec30704a13fe127`

• Glibc (2.6.1) - 15,398 KB:
Página web: *http://www.gnu.org/software/libc/*
Descarga: *http://ftp.gnu.org/gnu/glibc/glibc-2.6.1.tar.bz2*
Súma MD5: `11cf6d3fc86dbe0890b8d00372eb6286`

• Glibc LibIDN add-on (2.6.1) - 100 KB:
Descarga: *http://ftp.gnu.org/gnu/glibc/glibc-libidn-2.6.1.tar.bz2*
Súma MD5: `503f1315afd808728ebaa75b3d87a7d9`

• Grep (2.5.1a) - 516 KB:
Página web: *http://www.gnu.org/software/grep/*
Descarga: *http://ftp.gnu.org/gnu/grep/grep-2.5.1a.tar.bz2*
Súma MD5: `52202fe462770fa6be1bb667bd6cf30c`

• Groff (1.18.1.4) - 2,265 KB:
Página web: *http://www.gnu.org/software/groff/*
Descarga: *http://ftp.gnu.org/gnu/groff/groff-1.18.1.4.tar.gz*
Súma MD5: `ceecb81533936d251ed015f40e5f7287`

• GRUB (0.97) - 950 KB:
Página web: *http://www.gnu.org/software/grub/*
Descarga: *ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz*
Súma MD5: `cd3f3eb54446be6003156158d51f4884`

• Gzip (1.3.12) - 451 KB:
Página web: *http://www.gzip.org/*
Descarga: *http://ftp.gnu.org/gnu/gzip/gzip-1.3.12.tar.gz*
Súma MD5: `b5bac2d21840ae077e0217bc5e4845b1`

• Iana-Etc (2.20) - 191 KB:
Página web: *http://www.sethwklein.net/projects/iana-etc/*
Descarga: *http://www.sethwklein.net/projects/iana-etc/downloads/iana-etc-2.20.tar.bz2*
Súma MD5: `51d584b7b6115528c21e8ea32250f2b1`

• Inetutils (1.5) - 1,357 KB:
Página web: *http://www.gnu.org/software/inetutils/*
Descarga: *http://ftp.gnu.org/gnu/inetutils/inetutils-1.5.tar.gz*
Súma MD5: `aeacd11d19bf25c89d4eff38346bdfb9`

• IPRoute2 (2.6.20-070313) - 394 KB:
Página web: *http://linux-net.osdl.org/index.php/Iproute2*
Descarga: *http://developer.osdl.org/dev/iproute2/download/iproute2-2.6.20-070313.tar.gz*
Súma MD5: `7bc5883aadf740761fa2dd70b661e8cc`

• Kbd (1.12) - 618 KB:
Descarga: *http://www.kernel.org/pub/linux/utils/kbd/kbd-1.12.tar.bz2*
Súma MD5: `069d1175b4891343b107a8ac2b4a39f6`

• Less (406) - 285 KB:
Página web: *http://www.greenwoodsoftware.com/less/*
Descarga: *http://www.greenwoodsoftware.com/less/less-406.tar.gz*
Súma MD5: `c6062663b5be92dfcdfd6300ba0811e4`

• LFS-Bootscripts (20070813) - 39 KB:
Descarga: *http://www.linuxfromscratch.org/lfs/downloads/development/lfs-bootscripts-20070813.tar.bz2*
Súma MD5: `0ecbdd3b774d519fc535a0a595aa5b86`

• Libtool (1.5.24) - 2,851 KB:
Página web: *http://www.gnu.org/software/libtool/*
Descarga: *http://ftp.gnu.org/gnu/libtool/libtool-1.5.24.tar.gz*
Súma MD5: `d0071c890101fcf4f2be8934a37841b0`

• Linux (2.6.22.6) - 44,052 KB:
Página web: *http://www.kernel.org/*
Descarga: *http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.22.6.tar.bz2*
Súma MD5: `20af4d1e05bd725e89b691da483276e9`

Nota

El núcleo Linux se actualiza con relativa frecuencia, en muchas ocasiones debido al descubrimiento de vulneraciones de seguridad. Debe usarse la última version del núcleo 2.6.22.x disponible, a no ser que la página de erratas diga otra cosa.

• M4 (1.4.10) - 722 KB:
Página web: *http://www.gnu.org/software/m4/*
Descarga: *http://ftp.gnu.org/gnu/m4/m4-1.4.10.tar.bz2*
Súma MD5: `0a35bab2f5d605e08083d7e3cbd4b8b0`

• Make (3.81) - 1,125 KB:
Página web: *http://www.gnu.org/software/make/*
Descarga: *http://ftp.gnu.org/gnu/make/make-3.81.tar.bz2*
Súma MD5: `354853e0b2da90c527e35aabb8d6f1e6`

• Man-DB (2.4.4) - 877 KB:
Página web: *http://www.nongnu.org/man-db/*
Descarga: *http://savannah.nongnu.org/download/man-db/man-db-2.4.4.tar.gz*
Súma MD5: `9d7952de1aeb9121497a8204c59c56d7`

• Man-pages (2.64) - 1,799 KB:
Descarga: *http://www.kernel.org/pub/linux/docs/manpages/Archive/man-pages-2.64.tar.bz2*
Súma MD5: `80683dd76cd4eb5aed36ac4401aef4d6`

• Mktemp (1.5) - 69 KB:
Página web: *http://www.mktemp.org/*
Descarga: *ftp://ftp.mktemp.org/pub/mktemp/mktemp-1.5.tar.gz*
Súma MD5: `9a35c59502a228c6ce2be025fc6e3ff2`

• Module-Init-Tools (3.2.2) - 166 KB:
Página web: *http://www.kerneltools.org/*
Descarga: *http://www.kerneltools.org/pub/downloads/module-init-tools/module-init-tools-3.2.2.tar.bz2*
Súma MD5: `a1ad0a09d3231673f70d631f3f5040e9`

• Ncurses (5.6) - 2,346 KB:
Página web: *http://dickey.his.com/ncurses/*
Descarga: *ftp://invisible-island.net/ncurses/ncurses-5.6.tar.gz*
Súma MD5: `b6593abe1089d6aab1551c105c9300e3`

• Patch (2.5.4) - 183 KB:
Página web: *http://www.gnu.org/software/patch/*
Descarga: *http://ftp.gnu.org/gnu/patch/patch-2.5.4.tar.gz*
Súma MD5: `ee5ae84d115f051d87fcaaef3b4ae782`

• Perl (5.8.8) - 9,887 KB:
Página web: *http://www.perl.com/*
Descarga: *http://ftp.funet.fi/pub/CPAN/src/perl-5.8.8.tar.bz2*
Súma MD5: `a377c0c67ab43fd96eeec29ce19e8382`

• Procps (3.2.7) - 275 KB:
Página web: *http://procps.sourceforge.net/*
Descarga: *http://procps.sourceforge.net/procps-3.2.7.tar.gz*
Súma MD5: `f490bca772b16472962c7b9f23b1e97d`

• Psmisc (22.5) - 271 KB:
Página web: *http://psmisc.sourceforge.net/*
Descarga: *http://prdownloads.sourceforge.net/psmisc/psmisc-22.5.tar.gz*
Súma MD5: `09c20fd899c2c1bd2dce02a510f99fab`

• Readline (5.2) - 1,990 KB:
Página web: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*
Descarga: *http://ftp.gnu.org/gnu/readline/readline-5.2.tar.gz*
Súma MD5: `e39331f32ad14009b9ff49cc10c5e751`

• Sed (4.1.5) - 781 KB:
Página web: *http://www.gnu.org/software/sed/*
Descarga: *http://ftp.gnu.org/gnu/sed/sed-4.1.5.tar.gz*
Súma MD5: `7a1cbbbb3341287308e140bd4834c3ba`

• Shadow (4.0.18.1) - 1,481 KB:
Descarga: *http://anduin.linuxfromscratch.org/sources/LFS/lfs-packages/development/shadow-4.0.18.1.tar.bz2*
Súma MD5: `e7751d46ecf219c07ae0b028ab3335c6`

• Sysklogd (1.5) - 85 KB:
Página web: *http://www.infodrom.org/projects/sysklogd/*
Descarga: *http://www.infodrom.org/projects/sysklogd/download/sysklogd-1.5.tar.gz*
Súma MD5: `e053094e8103165f98ddafe828f6ae4b`

• Sysvinit (2.86) - 97 KB:
Descarga: *ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/sysvinit-2.86.tar.gz*
Súma MD5: `7d5d61c026122ab791ac04c8a84db967`

• Tar (1.18) - 1,833 KB:
Página web: *http://www.gnu.org/software/tar/*
Descarga: *http://ftp.gnu.org/gnu/tar/tar-1.18.tar.bz2*
Súma MD5: `70170208d7c1bb9ab40120579434b6a3`

• Tcl (8.4.15) - 3,549 KB:
Página web: *http://tcl.sourceforge.net/*
Descarga: *http://prdownloads.sourceforge.net/tcl/tcl8.4.15-src.tar.gz*
Súma MD5: `5e1b71eef1f75a294072aa3218f62b66`

• Texinfo (4.9) - 1,489 KB:
Página web: *http://www.gnu.org/software/texinfo/*
Descarga: *http://ftp.gnu.org/gnu/texinfo/texinfo-4.9.tar.bz2*
Súma MD5: `f4458e4b81e5469fa0815c35654141ab`

• Udev (113) - 191 KB:
Página web: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html*
Descarga: *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-113.tar.bz2*
Súma MD5: `cb9a227206b9d85ae8cfc88fc51c1710`

• Udev Configuration Tarball - 13 KB:
Descarga: *http://www.linuxfromscratch.org/lfs/downloads/development/udev-config-20070731.tar.bz2*
Súma MD5: `49c72e712f38c18884bd11a9a3b7e968`

• Util-linux (2.12r) - 1,339 KB:
Descarga: *http://www.kernel.org/pub/linux/utils/util-linux/util-linux-2.12r.tar.bz2*
Súma MD5: `af9d9e03038481fbf79ea3ac33f116f9`

• Vim (7.1) - 6,714 KB:
Página web: *http://www.vim.org*
Descarga: *ftp://ftp.vim.org/pub/vim/unix/vim-7.1.tar.bz2*
Súma MD5: `44c6b4914f38d6f9aa959640b89da329`

• Vim (7.1) language files (optional) - 1,161 KB:
Página web: *http://www.vim.org*
Descarga: *ftp://ftp.vim.org/pub/vim/extra/vim-7.1-lang.tar.gz*
Súma MD5: `144aa049ba70621acf4247f0459f3ee7`

• Zlib (1.2.3) - 485 KB:
Página web: *http://www.zlib.net/*
Descarga: *http://www.zlib.net/zlib-1.2.3.tar.gz*
Súma MD5: `debc62758716a169df9f62e6ab2bc634`

Tamaño total de estos paquetes: 205 MB

3.3. Parches necesarios

Aparte de los paquetes, también se necesitan varios parches. Estos parches corrigen pequeños errores en los paquetes que debería solucionar su desarrollador. Los parches también hacen pequeñas modificaciones para facilitar el trabajo con el paquete. Los siguientes parches son necesarios para construir un sistema LFS:

• Bash Upstream Fixes Patch - 24 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/bash-3.2-fixes-6.patch*
Súma MD5: `b81a6b53cd361b3b2291a762dcba87d0`

• Bzip2 Documentation Patch - 1.6 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/bzip2-1.0.4-install_docs-1.patch*
Súma MD5: `6a5ac7e89b791aae556de0f745916f7f`

• Coreutils Internationalization Fixes Patch - 101 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/coreutils-6.9-i18n-1.patch*
Súma MD5: `806ce5bcb16a763a77bea411ec5ff637`

• Coreutils Suppress Uptime, Kill, Su Patch - 13 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/coreutils-6.9-suppress_uptime_kill_su-1.patch*
Súma MD5: `e8ae92cdec364ca2a318f5c4c77bf032`

• Coreutils Uname Patch - 4.6 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/coreutils-6.9-uname-1.patch*
Súma MD5: `c05b735710fbd62239588c07084852a0`

• Diffutils Internationalization Fixes Patch - 18 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/diffutils-2.8.1-i18n-1.patch*
Súma MD5: `c8d481223db274a33b121fb8c25af9f7`

• Expect Spawn Patch - 6.8 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/expect-5.43.0-spawn-1.patch*
Súma MD5: `ef6d0d0221c571fb420afb7033b3bbba`

• Gawk Segfault Patch - 1.3 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/gawk-3.1.5-segfault_fix-1.patch*
Súma MD5: `7679530d88bf3eb56c42eb6aba342ddb`

• Grep RedHat Fixes Patch - 55 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/grep-2.5.1a-redhat_fixes-2.patch*
Súma MD5: `2c67910be2d0a54714f63ce350e6d8a6`

• Groff Debian Patch - 379 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/groff-1.18.1.4-debian_fixes-1.patch*
Súma MD5: `05607e7fcfd6e5091f020bf44ddca10b`

• GRUB Disk Geometry Patch - 28 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/grub-0.97-disk_geometry-1.patch*
Súma MD5: `bf1594e82940e25d089feca74c6f1879`

• Inetutils No-Server-Man-Pages Patch - 5.3 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/inetutils-1.5-no_server_man_pages-2.patch*
Súma MD5: `ec83aa00fb111f6f9d9aca04de9cb753`

• Kbd Backspace/Delete Fix Patch - 11 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/kbd-1.12-backspace-1.patch*
Súma MD5: `692c88bb76906d99cc20446fadfb6499`

• Kbd GCC-4.x Fix Patch - 1.4 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/kbd-1.12-gcc4_fixes-1.patch*
Súma MD5: `615bc1e381ab646f04d8045751ed1f69`

• Man-DB Fix Patch - 2.0 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/man-db-2.4.4-fixes-1.patch*
Súma MD5: `f75b3c44bb801778cf188b87454ff9c1`

• Mktemp Tempfile Patch - 3.5 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/mktemp-1.5-add_tempfile-3.patch*
Súma MD5: `65d73faabe3f637ad79853b460d30a19`

• Module-init-tools Patch - 1.2 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/module-init-tools-3.2.2-modprobe-1.patch*
Súma MD5: `f1e452fdf3b8d7ef60148125e390c3e8`

• Ncurses Coverity Patch - 16.8 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/ncurses-5.6-coverity_fixes-1.patch*
Súma MD5: `aa2fa9d0e89bbfdb4ce7e0e6b4b46670`

• Perl Libc Patch - 1.1 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/perl-5.8.8-libc-2.patch*
Súma MD5: `3bf8aef1fb6eb6110405e699e4141f99`

• Readline Fixes Patch - 12.5 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/readline-5.2-fixes-4.patch*
Súma MD5: `4da6bf2066a7603c7bb0ab1f52243316`

• Shadow Useradd Patch - 6.1 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/shadow-4.0.18.1-useradd_fix-2.patch*
Súma MD5: `5f35528f38d5432d5fa2dd79d04bdfdd`

• Texinfo Multibyte Fixes Patch - 1.5 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/texinfo-4.9-multibyte-1.patch*
Súma MD5: `6cb5b760cfdd2dd53a0430eb572a8aaa`

• Texinfo Tempfile Fix Patch - 2.2 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/texinfo-4.9-tempfile_fix-1.patch*
Súma MD5: `559bda136a2ac7777ecb67511227af85`

• Util-linux Cramfs Patch - 2.8 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/util-linux-2.12r-cramfs-1.patch*
Súma MD5: `1c3f40b30e12738eb7b66a35b7374572`

• Util-linux Lseek Patch - 10 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/util-linux-2.12r-lseek-1.patch*
Súma MD5: `5d6c86321c1ea74d7ed7cf57861da423`

• Vim Fixes Patch - 294 KB:
Download: *http://www.linuxfromscratch.org/patches/lfs/development/vim-7.1-fixes-2.patch*
MD5 sum: `4b33bda20c4e75601cd581b745477a2d`

• Vim Man Directories Patch - 4.2 KB:
Descarga: *http://www.linuxfromscratch.org/patches/lfs/development/vim-7.1-mandir-1.patch*
Súma MD5: `b6426eb4192faba1e867ddd502323f5b`

Tamaño total de estos parches: 1,007.9 KB

Aparte de los anteriores parches necesarios, hay una serie de parches opcionales creados por la comunidad LFS. Estos parches opcionales solucionan pequeños problemas, o activan alguna funcionalidad que no lo está por defecto. Eres libre de examinar la base de datos de parches que se encuentra en *http://www.linuxfromscratch.org/patches* y elegir cualquier parche adicional que cubra las necesidades del sistema.

Capítulo 4. Últimos preparativos

4.1. Sobre $LFS

Durante este libro la variable de entorno `LFS` se usará frecuentemente. Es importante que esta variable esté siempre definida. Debería establecerse al punto de montaje que elegiste para tu partición LFS. Comprueba que tu variable `LFS` está correctamente establecida con:

```
echo $LFS
```

Asegúrate de que la salida muestra la ruta a tu punto de montaje de la partición LFS, que es `/mnt/lfs` si seguiste el ejemplo aquí usado. Si la salida es errónea, puedes establecer la variable con:

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

Tener establecida esta variable significa que si se indica que ejecutes un comando como **mkdir $LFS/tools**, puede teclearse literalmente. El intérprete de comandos sustituirá "$LFS" con "/mnt/lfs" (o aquello a lo que esté establecida la variable) cuando procese la línea de comandos.

No olvides comprobar que `$LFS` esté establecida cada vez que salgas y vuelvas al entorno (o cuando hagas **su** a `root` o a otro usuario).

4.2. Creación del directorio $LFS/tools

Todos los programas compilados en el Capítulo 5 se instalarán bajo `$LFS/tools` para mantenerlos separados de los programas compilados en el Capítulo 6. Los programas compilados aquí son sólo herramientas temporales y no formarán parte del sistema LFS final. Al mantener estos programas en un directorio aparte podrán eliminarse fácilmente tras su uso. Esto también evita que acaben en los directorios de producción de tu anfitrión (que es fácil que ocurra por accidente en el Capítulo 5).

Crea el directorio necesario ejecutando lo siguiente como `root`:

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

El próximo paso es crear un enlace `/tools` en el sistema anfitrión. Este apuntará al directorio que acabamos de crear en la partición LFS. Ejecuta este comando también como `root`:

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

Nota

El comando anterior es correcto. El comando **ln** tiene bastantes variaciones de sintaxis, por lo que asegúrate de comprobar **info coreutils ln** y `ln(1)` antes de informar de lo que puedes pensar que es un error.

El enlace simbólico creado posibilita que el conjunto de herramientas se compile siempre en referencia a `/tools`, de forma que el compilador, ensamblador y enlazador funcionarán en este capítulo (en el que todavía estamos utilizando algunas herramientas del sistema anfitrión) y en el siguiente (cuando "cambiemos la raíz" a la partición LFS).

4.3. Añadir el usuario lfs

Si se trabaja como `root`, un simple error puede dañar o destruir un sistema. Por tanto recomendamos construir los paquetes del siguiente capítulo como un usuario sin privilegios. Puedes usar tu propio nombre de usuario, pero para facilitar la creación de un entorno de trabajo limpio, crea un nuevo usuario llamado `lfs` como miembro de un nuevo grupo (llamado también `lfs`) y utilízalo para el proceso de construcción. Como `root`, ejecuta el siguiente comando para añadir el nuevo usuario:

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

Significado de las opciones:

`-s /bin/bash`
: Esto hace de **bash** el intérprete de comandos por defecto para el usuario `lfs`.

`-g lfs`
: Esta opción añade el usuario `lfs` al grupo `lfs`.

`-m`
: Esto crea el directorio personal para `lfs`.

`-k /dev/null`
: Este parámetro evita que se copien ficheros procedentes de un posible esqueleto de directorio (por defecto es `/etc/skel`), cambiando la localización de entrada al dispositivo especial nulo.

`lfs`
: Este es el nombre real del usuario y grupo creados.

Para ingresar como `lfs` (en vez de cambiar al usuario `lfs` cuando se está como `root`, que no precisa que el usuario `lfs` tenga una contraseña), asígnale una contraseña a `lfs`:

```
passwd lfs
```

Concede a `lfs` acceso completo a `$LFS/tools` dándole la propiedad del directorio:

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

Si creaste un directorio de trabajo como te sugerimos, haz que el usuario `lfs` sea también el propietario de este directorio:

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

A continuación, entra como usuario `lfs`. Esto se puede hacer mediante una consola virtual, con un administrador de sesión gráfico o con el siguiente comando de sustitución de usuario:

```
su - lfs
```

El "`-`" le indica a **su** que inicie un intérprete de comandos de ingreso, en lugar de uno de no ingreso. La diferencia entre estos dos tipos de intérpretes de comandos se encuentra detallada en `bash(1)` e **info bash**.

4.4. Configuración del entorno

Establece un buen entorno de trabajo mediante la creación de dos nuevos ficheros de inicio para el intérprete de comandos **bash**. Estando en el sistema como usuario `lfs`, ejecuta los siguientes comandos para crear un `.bash_profile` nuevo:

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

Cuando entras como usuario `lfs` el intérprete de comandos inicial es un intérprete *de ingreso* que lee el `/etc/profile` de tu anfitrión (que posiblemente contenga algunos ajustes de variables de entorno) y luego lee `.bash_profile`. El comando **exec env -i.../bin/bash** del fichero `.bash_profile` sustituye el intérprete de comandos en ejecución por uno nuevo con un entorno completamente vacío, excepto por las variables `HOME`, `TERM` y `PS1`. Esto asegura que en el entorno de construcción no aparezcan variables de entorno indeseadas o dañinas procedentes del sistema anfitrión. La técnica aquí usada consigue el objetivo de asegurar un entorno limpio.

La nueva instancia del intérprete comandos es un intérprete de *no ingreso* que no lee los ficheros `/etc/profile` o `.bash_profile`, pero en su lugar lee el fichero `.bashrc`. Crea ahora el fichero `.bashrc`:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
PATH=/tools/bin:/bin:/usr/bin
export LFS LC_ALL PATH
EOF
```

El comando **set +h** desactiva la función de tablas de dispersión (hash) de **bash**. Normalmente, esta función es muy útil: **bash** usa una tabla de dispersión para recordar la ruta completa de los ejecutables, evitando búsquedas reiteradas en el `PATH` para encontrar el mismo binario. Sin embargo, las nuevas herramientas deberían utilizarse a medida que son instaladas. Al desactivar esta característica, el intérprete de comandos siempre buscará en el `PATH` cuando deba ejecutarse un programa. Por tanto, el intérprete de comandos encontrará las herramientas recién compiladas en `$LFS/tools` tan pronto como estén disponibles, sin recordar una anterior versión del mismo programa en una ubicación diferente.

Establecer la máscara de creación de ficheros (umask) a 022 asegura que los ficheros y directorios de nueva creación sólo pueden ser escritos por su propietario, pero son legibles y ejecutables por cualquiera (asumiendo que los modos por defecto son usados por la llamada `open(2)` del sistema, los nuevos ficheros tendrán permisos 644 y los directorios 755).

La variable `LFS` debe establecerse al punto de montaje elegido.

La variable `LC_ALL` controla la localización de ciertos programas, haciendo que sus mensajes sigan las convenciones para un determinado país. Si el sistema anfitrión utiliza una versión de Glibc anterior a la 2.2.4, tener `LC_ALL` establecida a algo diferente a “POSIX” o “C” (durante el siguiente capítulo) puede causar problemas si sales del entorno chroot e intentas regresar más tarde. Establecer `LC_ALL` a “POSIX” o “C” (ambos son equivalentes) asegura que todo funcionará como se espera dentro del entorno chroot.

Al añadir `/tools/bin` al principio del `PATH`, todos los programas instalados en el Capítulo 5 son inmediatamente detectados por el intérprete de comandos tras su instalación. Esto, combinado con la desactivación de las tablas de dispersión, limita el riesgo de utilizar los antiguos programas del anfitrión cuando dichos pogramas ya están disponibles en el entorno del capítulo 5.

Finalmente, para tener el entorno preparado por completo para construir las herramientas temporales, carga el perfil de usuario recién creado:

```
source ~/.bash_profile
```

4.5. Sobre los SBUs

Bastante gente desea saber de antemano cuanto tiempo, aproximadamente, le llevará compilar e instalar cada paquete. Pero Linux From Scratch puede construirse sobre muchos sistemas diferentes, siendo imposible dar tiempos reales y precisos. El paquete más grande (Glibc) tardará unos 20 minutos en un sistema rápido, ¡pero puede tardar hasta tres días en uno lento! Así que en vez de proporcionar tiempos reales se usará la unidad de medida Standard Build Unit (SBU, Unidad de Construcción Estandar).

Funciona de la siguiente forma. El primer paquete que se compila en este libro es, en el Capítulo 5, Binutils. El tiempo que tarde en compilar este paquete es lo que llamamos Unidad de Construcción Estandar o SBU. Todos los demás tiempos de compilación se expresarán con relación a este tiempo.

Por ejemplo, considera un paquete cuyo tiempo de compilación es de 4,5 SBUs. Esto significa que si un sistema tarda en compilar e instalar el primer paso de Binutils 10 minutos, tardará *aproximadamente* 45 minutos en construir dicho paquete. Por suerte, bastantes de los tiempos de construcción son mucho más cortos que el de Binutils.

En general, los SBU no son muy exactos debido a que dependen de muchos factores, no sólo la versión de GCC del anfitrión. Advierte que en máquinas basadas en Multiprocesadores Simétricos (SMP) los SBU son aún menos exactos. Se han suministrado aquí para mostrar una aproximación de cuanto podría tardar la instalación de un paquete, pero los números pueden variar en docenas de minutos en algunos casos.

Para ver los tiempos reales de un cierto número de máquinas concretas, recomendamos visitar "The LinuxFromScratch SBU Home Page", que se encuentra en *http://www.linuxfromscratch.org/~sbu/*.

4.6. Sobre los bancos de pruebas

Muchos paquetes proporcionan un banco de pruebas. Ejecutar el banco de pruebas para un paquete recién construido es una buena idea, pues puede proporcionar una “verificación de calidad” indicando que todo se ha compilado correctamente. Un banco de pruebas que supere sus comprobaciones normalmente confirma que el paquete está funcionando tal y como el desarrollador espera. Pero esto, sin embargo, no garantiza que el paquete está totalmente libre de errores.

Algunos bancos de pruebas son más importantes que otros. Por ejemplo, los bancos de pruebas de los paquetes de las herramientas principales (GCC, Binutils y Glibc) son de la mayor importancia debido a su papel central en el correcto funcionamiento del sistema. Los bancos de pruebas para GCC y Glibc pueden tardar bastante tiempo en completarse, sobre todo en hardware lento, pero son muy recomendables.

Nota

La experiencia ha mostrado que se gana poco ejecutando los bancos de pruebas en el Capítulo 5. No se puede escapar del hecho de que el sistema anfitrión siempre ejerce cierta influencia sobre las pruebas en dicho capítulo, causando con frecuencia fallos inexplicables. Debido a que las herramientas construidas en el Capítulo 5 son temporales y descartables, no recomendamos el lector medio ejecutar los bancos de pruebas en el Capítulo 5. Las instrucciones para ejecutarlos se suministran en benficio de los verificadores y desarrolladores, pero son estrictamente opcionales.

Un problema común al ejecutar los bancos de pruebas de Binutils y GCC es quedarse sin seudo-terminales (PTYs). El síntoma es un número inusualmente alto de pruebas fallidas. Esto puede suceder por diferentes razones, pero lo más probable es que el sistema anfitrión no tenga el sistema de ficheros `devpts` configurado correctamente. En el Capítulo 5 se tratará este tema con mayor detalle.

En ocasiones los bancos de pruebas de los paquetes muestran falsos fallos, pero por razones conocidas por los desarrolladores y que no consideran críticas. Consulta los registros que se encuentran en *http://www.linuxfromscratch.org/lfs/build-logs/development/* para verificar si estos fallos son normales o no. Este sitio es válido para todas las pruebas que aparecen en el libro.

Capítulo 5. Construir un sistema temporal

5.1. Introducción

Este capítulo muestra cómo compilar e instalar un sistema Linux mínimo. Este sistema contendrá sólo las herramientas necesarias para poder iniciar la construcción del sistema LFS definitivo en el Capítulo 6, permitiendo un entorno de trabajo algo más amigable para el usuario que el que un entorno mínimo ofrecería.

La construcción de este sistema minimalista se hará en dos etapas. La primera es construir un conjunto de herramientas independiente del sistema anfitrión (compilador, ensamblador, enlazador, librerías y unas pocas herramientas útiles). La segunda etapa utiliza estas herramientas para construir el resto de herramientas esenciales.

Los ficheros compilados en este capítulo se instalarán bajo el directorio `$LFS/tools` para mantenerlos separados de los ficheros que se instalen en el siguiente capítulo y de los directorios de producción de tu anfitrión. Puesto que los paquetes compilados aquí son puramente temporales, no queremos que estos ficheros contaminen el futuro sistema LFS.

Importante

Antes de ejecutar las instrucciones de construcción de un paquete, debes desempaquetarlo como usuario `lfs` y hacer un **`cd`** para entrar al directorio creado. Las instrucciones de construcción asumen que estás usando el intérprete de comandos **bash**.

Varios de los paquetes deben parchearse antes de compilarlos, pero sólo cuando el parche es necesario para solucionar un problema. Con frecuencia el parche es necesario tanto en éste como en el siguiente capítulo, pero a veces sólo es necesario en uno de ellos. Por lo tanto, no te preocupes si parece que faltan las instrucciones para uno de los parches descargados. Igualmente, cuando se aplique un parche ocasionalmente verás un mensaje de aviso sobre *offset* o *fuzz*. No debes preocuparte por estos avisos, pues el parche se aplicará correctamente.

Durante la compilación de muchos paquetes verás aparecer en pantalla diversos avisos (warnings). Esto es normal y puedes ignorarlos con tranquilidad. No son más que eso, avisos; la mayoría debidos a un uso inapropiado, pero no inválido, de la sintaxis de C o C++. Se debe a que los estándares de C cambian con frecuencia y algunos paquetes todavía usan un estándar antiguo. Esto no es un problema, pero hace que se muestre el aviso.

Importante

Tras instalar cada paquete debes borrar sus directorios de fuentes y de construcción, excepto si se indica lo contrario. Borrar las fuentes evita fallos de configuración cuando el mismo paquete se reinstale más adelante.

Comprueba de nuevo que la variable de entorno `LFS` está correctamente establecida:

```
echo $LFS
```

Asegúrate de que la salida muestra la ruta al punto de montaje de tu partición LFS, que es `/mnt/lfs` si seguiste nuestro ejemplo.

5.2. Notas técnicas sobre las herramientas

Esta sección explica algunos de los razonamientos y detalles técnicos que hay detrás del sistema de construcción. No es esencial que entiendas todo esto inmediatamente. La mayor parte tendrá sentido cuando hayas hecho una construcción real. Puedes consultar esta sección en cualquier momento durante la construcción.

El principal objetivo del Capítulo 5 es proporcionar un entorno temporal al que podamos entrar con chroot y a partir del cual podamos generar una construcción limpia y libre de problemas del sistema LFS en el Capítulo 6. Por el camino intentaremos independizarnos todo lo posible del sistema anfitrión, y para eso construimos unas herramientas principales autocontenidas y autohospedadas. Debería tenerse en cuenta que el proceso de construcción ha sido diseñado de forma que se minimice el riesgo para los nuevos lectores y, al mismo tiempo, se proporcione el máximo valor educacional.

Importante

Antes de continuar, deberías informarte del nombre de tu plataforma de trabajo, conocido con frecuencia como *target triplet* (triplete del objetivo). Para muchos el "target triplet" posiblemente sea *i686-pc-linux-gnu*. Una forma simple de determinar tu "target triplet" es ejecutar el guión **config.guess** que se incluye con las fuentes de muchos paquetes. Desempaqueta las fuentes de Binutils, ejecuta el guión **`./config.guess`** y anota el resultado.

Igualmente necesitarás saber el nombre del enlazador dinámico de tu plataforma, también conocido como cargador dinámico (no debe confundirse con el enlazador estándar **ld**, que es parte de Binutils). El enlazador dinámico suministrado por Glibc encuentra y carga las librerías compartidas necesarias para un programa, prepara el programa y lo ejecuta. Usualmente el nombre del enlazador dinámico es `ld-linux.so.2`. En plataformas menos conocidas puede ser `ld.so.1` y en las nuevas plataformas de 64 bits puede que incluso sea algo totalmente diferente. El nombre del enlazador dinámico de tu plataforma puede determinarse mirando en el directorio `/lib` de tu sistema anfitrión. Un modo seguro es inspeccionar un binario cualquiera de tu sistema anfitrión ejecutando: **`readelf -l <nombre del binario> | grep interpreter`** y anotar la salida. La referencia autorizada que cubre todas las plataformas está en el fichero `shlib-versions` en la raíz del árbol de las fuentes de Glibc.

Algunas claves técnicas sobre cómo funciona el método de construcción del Capítulo 5:

- Similar en principio a la compilación cruzada, donde las herramientas instaladas dentro del mismo prefijo trabajan en cooperación y utilizan una pequeña "magia" de GNU.
- Cuidada manipulación de la ruta de búsqueda de librerías del enlazador estándar para asegurar que los programas se enlazan sólo contra las librerías que elegimos.
- Cuidada manipulación del fichero `specs` de **gcc** para indicarle al compilador cuál es el enlazador dinámico a usar.

Se instala primero Binutils debido a que, tanto en GCC como en Glibc, la ejecución de **configure** realiza varias pruebas sobre el ensamblador y el enlazador para determinar qué características del software deben activarse o desactivarse. Esto es más importante de lo que uno podría pensar. Un GCC o una Glibc incorrectamente configurados puede provocar unas herramientas sutilmente rotas cuyo impacto podría no notarse hasta casi finalizada la construcción de una distribución completa. Un fallo en el banco de pruebas normalmente resaltará dicho error antes de perder demasiado tiempo.

Binutils instala tanto su ensamblador como su enlazador en dos ubicaciones, `/tools/bin` y `/tools/$TARGET_TRIPLET/bin`. Las herramientas de una ubicación son enlaces duros a la otra. Un aspecto importante del enlazador es su orden de búsqueda de librerías. Puede obtenerse información detallada de **ld** pasándole la opción *`--verbose`*. Por ejemplo, un **ld --verbose | grep SEARCH** mostrará las rutas de búsqueda actuales y su orden. Puedes ver qué ficheros son realmente enlazados por **ld** compilando un programa simulado y pasándole la opción *`--verbose`*. Por ejemplo, **`gcc dummy.c -Wl,--verbose 2>&1 | grep succeeded`** te mostrará todos los ficheros abiertos con éxito durante el enlazado.

El siguiente paquete instalado es GCC y durante su fase **configure** verás, por ejemplo:

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

comprobando qué ensamblador usar...
    /tools/i686-pc-linux-gnu/bin/as
comprobando qué enlazador usar...
    /tools/i686-pc-linux-gnu/bin/ld
```

Esto es importante por la razón mencionada antes. También demuestra que el guión configure de GCC no explora los directorios del PATH para encontrar las herramientas a usar. Sin embargo, durante la operación real del propio **gcc**, no se utilizan necesariamente las mismas rutas de búsqueda. Para saber cuál es el enlazador estándar que utilizará **gcc**, ejecuta: **`gcc -print-prog-name=ld`**.

Puedes obtener información detallada a partir de **gcc** pasándole la opción *`-v`* mientras compilas un programa simulado. Por ejemplo: **`gcc -v dummy.c`** te mostrará los detalles sobre las fases de preprocesamiento, compilación y ensamblado, incluidas las rutas de búsqueda de **gcc** y su orden.

A continuación se instala Glibc. Las consideraciones más importantes para la construcción de Glibc son el compilador, las herramientas de binarios y las cabeceras del núcleo. Normalmente el compilador no es problema, pues Glibc siempre utilizará el **gcc** que se encuentre en un directorio del `PATH`. Las herramientas de binarios y las cabeceras del núcleo pueden ser algo mas problemáticas, así que no nos arriesgaremos y haremos uso de las opciones disponibles de configure para forzar las opciones correctas. Después de ejecutar **configure** puedes revisar el contenido del fichero `config.make` en el directorio `glibc-build` para ver todos los detalles importantes. Encontrarás algunas cosas interesantes, como el uso de *`CC="gcc -B/tools/bin/"`* para controlar qué herramientas de binarios son usadas, y también el uso de las opciones *`-nostdinc`* y *`-isystem`* para controlar la ruta de búsqueda de cabeceras del compilador. Estos detalles ayudan a resaltar un aspecto importante del paquete Glibc: es muy autosuficiente en cuanto a su maquinaria de construcción y generalmente no se apoya en las opciones por defecto de las herramientas.

Después de la instalación de Glibc, haremos algunos ajustes para asegurar que la búsqueda y el enlazado tengan lugar solamente dentro de nuestro directorio `/tools`. Instalaremos un **ld** ajustado, que tiene limitada su ruta de búsqueda interna a `/tools/lib`. Entonces retocaremos el fichero specs de **gcc** para que apunte a nuestro nuevo enlazador dinámico en `/tools/lib`. Este último paso es vital para el proceso completo. Como se mencionó antes, dentro de cada ejecutable compartido ELF se fija la ruta a un enlazador dinámico. Puedes verificar esto mediante: **`readelf -l <nombre del binario> | grep interpreter`**. Retocando el ficheros specs de **gcc** estaremos seguros de que todo binario compilado desde aquí hasta el final de este capítulo usará nuestro nuevo enlazador dinámico en `/tools/lib`.

La necesidad de utilizar el nuevo enlazador dinámico es también la razón por la que aplicamos el parche Specs en la segunda fase de GCC. De no hacer esto los propios programas de GCC incluirían dentro suyo el nombre del enlazador dinámico del directorio `/lib` del sistema anfitrión, lo que arruinaría nuestro objetivo de librarnos del anfitrión.

Durante la segunda fase de Binutils podremos usar la opción *`--with-lib-path`* de configure para controlar la ruta de búsqueda de librerías de **ld**. A partir de este punto el corazón de las herramientas está autocontenido y autohospedado. El resto de los paquetes del Capítulo 5 se construirán todos contra la nueva Glibc en `/tools`.

Tras entrar en el entorno chroot en el Capítulo 6, el primer gran paquete a instalar es Glibc, debido a su naturaleza autosuficiente. Una vez que esta Glibc se instale dentro de `/usr`, haremos un rápido cambio en las opciones por defecto de las herramientas y entonces procederemos a la construcción real del sistema LFS.

5.3. Binutils-2.18 - Fase 1

El paquete Binutils contiene un enlazador, un ensamblador y otras utilidades para trabajar con ficheros objeto.

Tiempo estimado de construcción:	1 SBU
Espacio requerido en disco:	213 MB

5.3.1. Instalación de Binutils

Es importante que Binutils sea el primer paquete que compiles, pues tanto Glibc como GCC llevan a cabo varias comprobaciones sobre el enlazador y el ensamblador disponibles para determinar qué características activar.

La documentación de Binutils recomienda construirlo en un directorio dedicado, fuera del árbol de las fuentes:

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

Nota

Si quieres que los valores de los SBUs mostrados en el resto del libro sean de utilidad, mide el tiempo que se tarda en construir este paquete desde la compilación hasta la primera instalación. Para ello, envuelve los comandos dentro de un comando **time** de esta forma: **`time { ./configure ... && make && make install; }`**.

Prepara Binutils para su compilación:

```
CC="gcc -B/usr/bin/" ../binutils-2.18/configure \
    --prefix=/tools --disable-nls --disable-werror
```

Significado de las opciones de configure:

`CC="gcc -B/usr/bin/"`
Esto fuerza que **gcc** prefiera el enlazador del anfitrión en `/usr/bin`. Esto es necesario con ciertos anfitriones en los que el nuevo **ld** construido aquí no es compatible con el **gcc** del anfitrión.

`--prefix=/tools`
Esto le indica al guión configure que los programas de Binutils se instalarán en el directorio `/tools`.

`--disable-nls`
Esta opción desactiva la internacionalización, pues i18n no es necesario en las herramientas temporales.

`--disable-werror`
Esto evita que la construcción se pare en el caso de que el compilador del anfitrión genere avisos.

Compila el paquete:

```
make
```

La compilación se ha completado. Normalmente deberíamos ejecutar ahora el banco de pruebas, pero en esta temprana fase el entorno de trabajo para los bancos de pruebas (Tcl, Expect y DejaGnu) todavía no está en su sitio. Los beneficios de ejecutar las pruebas ahora son mínimos, pues los programas de esta primera fase pronto serán sustituidos por los de la segunda.

Instala el paquete:

```
make install
```

Prepara el enlazador para la posterior fase de “ajuste”:

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

Significado de los parámetros de make:

`-C ld clean`
: Esto le indica al programa make que elimine todos los ficheros compilados que haya en el subdirectorio `ld`.

`-C ld LIB_PATH=/tools/lib`
: Esta opción vuelve a construir todo dentro del subdirectorio `ld`. Especificar la variable `LIB_PATH` del Makefile en la línea de comandos nos permite obviar su valor por defecto y apuntar a nuestro directorio de herramientas temporales. El valor de esta variable especifica la ruta de búsqueda de librerías por defecto del enlazador. Estos preparativos se utilizan más tarde en este capítulo.

Los detalles sobre este paquete se encuentran en la Sección 6.11.2, “Contenido de Binutils”.

5.4. GCC-4.2.1 - Fase 1

El paquete GCC contiene la colección de compiladores GNU, que incluye los compiladores C y C++.

Tiempo estimado de construcción:	9.2 SBU
Espacio requerido en disco:	655 MB

5.4.1. Instalación de GCC

La documentación de GCC recomienda construirlo en un directorio dedicado, fuera del árbol de las fuentes:

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

Prepara GCC para su compilación:

```
CC="gcc -B/usr/bin/" ../gcc-4.2.1/configure --prefix=/tools \
    --with-local-prefix=/tools --disable-nls --enable-shared \
    --enable-languages=c
```

Significado de las opciones de configure:

`CC="gcc -B/usr/bin/"`
Esto fuerza que **gcc** prefiera el enlazador del anfitrión en `/usr/bin`. Esto es necesario con ciertos anfitriones en los que el nuevo **ld** construido aquí no es compatible con el **gcc** del anfitrión.

`--with-local-prefix=/tools`
Esta opción es para eliminar `/usr/local/include` de las rutas de búsqueda por defecto de **gcc**. Esto no es esencial, sin embargo ayuda a minimizar la influencia del sistema anfitrión.

`--enable-shared`
Esta opción permite construir `libgcc_s.so.1` y `libgcc_eh.a`. Tener a `libgcc_eh.a` disponible nos asegura que el guión configure de Glibc (el siguiente paquete por compilar) produzca los resultados apropiados.

`--enable-languages=c`
Esta opción nos asegura que sólo se construya el compilador de C.

El siguiente comando compilará GCC no solo una vez, si no varias veces. Se usan los programas compilados la primera vez para compilarse as si mismo una segunda vez, y repite el proceso una tercera vez. Entonces compara estos segundo y tercero compiladores para asegurarse de que puede reproducirse a si mismo correctamente. Esto es conocido como “bootstrapping”. Construir GCC de este modo asegura que fué construido correctamente ys es ahora la configuración por defecto del paquete. Continua la compilación ejecutando:

```
make
```

La compilación se ha completado. En este punto normalmente ejecutaríamos el banco de pruebas, pero, como se mencionó antes, el entorno de trabajo para los bancos de pruebas no se encuentra todavía en su lugar. Los beneficios de ejecutar ahora los bancos de pruebas son mínimos, pues los programas de esta primera fase pronto serán sustituidos.

Instala el paquete:

```
make install
```

Como toque final, crea un enlace simbólico. Muchos programas y guiones ejecutan **cc** en vez de **gcc**. Esto es una forma de hacer que los programas sean genéricos y por tanto utilizables en toda clase de sistemas Unix. No todos tienen instalado el compilador de C de GNU. Ejecutar **cc** deja al administrador del sistema libre de decidir qué compilador de C instalar, mientras haya un enlace simbólico que apunte a él:

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

Los detalles sobre este paquete se encuentran en la Sección 6.12.2, “Contenido de GCC”.

5.5. Cabeceras API de Linux-2.6.22.6

Las cabeceras API de Linux exponen la API del núcleo para ser usada por Glibc.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 286 MB

5.5.1. Instalación de las cabeceras API de Linux

El núcleo Linux necesita exponer una Interfaz de Programación de Aplicaciones (API) para ser usada por la librería C del sistema (Glibc en LFS). Esto se hace sanitizando los diversos ficheros de cabecera incluidos en el paquete con las fuentes de núcleo Linux.

Instala los ficheros de cabecera:

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

Los detalles sobre este paquete se encuentran en la Sección 6.7.2, “Contenido de las cabeceras API de Linux”.

5.6. Glibc-2.6.1

El paquete Glibc contiene la librería C principal. Esta librería proporciona todas las rutinas básicas para la ubicación de memoria, búsqueda de directorios, abrir y cerrar ficheros, leerlos y escribirlos, manejo de cadenas, coincidencia de patrones, aritmética, etc...

Tiempo estimado de construcción: 7 SBU
Espacio requerido en disco: 342 MB

5.6.1. Instalación de Glibc

La documentación de Glibc recomienda construirlo fuera del árbol de las fuentes, en un directorio de construcción dedicado:

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

Debido a que Glibc ya no soporta i386, sus desarrolladores dicen que ha de usarse la opción de compilación *`-march=i486`* cuando se construye para máquinas x86. Hay varias formas de hacer esto, pero las pruebas nuestran que es mejor poner la opción dentro de la variable “CFLAGS”. En vez de sobreescribir por completo lo que el sistema interno de construcción de Glibc utiliza en CFLAGS, añadiremos la nueva opción al contenido existente mediante el uso del ficgero especial `configparms`:

```
echo "CFLAGS += -march=i486" > configparms
```

Prepara Glibc para su compilación:

```
../glibc-2.6.1/configure --prefix=/tools \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --with-binutils=/tools/bin \
    --without-gd --with-headers=/tools/include \
    --without-selinux
```

Significado de las opciones de configure:

`--disable-profile`
Esto construye las librerías sin información de perfiles. Omite esta opción si planeas usar perfiles en las herramientas temporales.

`--enable-add-ons`
Esto le indica a Glibc que utilice el añadido NPTL como su librería de hilos.

`--enable-kernel=2.6.0`
Esto le indica a Glibc que compile la librería con soporte para núcleos Linux 2.6.x.

`--with-binutils=/tools/bin`
Aunque no es necesario, esta opción nos asegura que no haya equívocos sobre qué programas de Binutils se utilizarán durante la construcción de Glibc.

`--without-gd`
Esto evita la construcción del programa **memusagestat**, el cual insiste en enlazarse contra las librerías del sistema anfitrión (libgd, libpng, libz y demás).

`--with-headers=/tools/include`
Esto le indica a Glibc que se compile contra las cabeceras recién instaladas en el directorio de herramientas, para que conozca exactamente las características que tiene el núcleo y pueda optimizarse correctamente.

`--without-selinux`
Cuando se construye a partir de un anfitrión que utiliza la funcionalidad de SELinux (como Fedora Core 3), Glibc se construirá con soporte para SELinux. Como las herramientas del entorno LFS no contienen soporte para SELinux, una Glibc compilada con dicho soporte no funcionará correctamente.

Durante esta fase puede que veas el siguiente mensaje de aviso:

```
configure: WARNING:
*** These auxiliary programs are missing or
*** incompatible versions: msgfmt
*** some features will be disabled.
*** Check the INSTALL file for required versions.

configure: AVISO:
*** Versión incompatible o ausente de estos
*** programas auxiliares: msgfmt
*** algunas características serán desactivadas.
*** Comprueba en el fichero INSTALL las versiones requeridas.
```

Normalmente, la ausencia o incompatibilidad del programa **msgfmt** es inofensiva, pero se cree que en ocasiones puede causar problemas al ejecutar el banco de pruebas. El programa **msgfmt** es parte del paquete Gettext y debería proporcionarlo el sistema anfitrión. Si **msgfmt** está presente pero es incompatible, actualiza el paquete Gettext del sistema anfitrión o continúa sin él y observa si los bancos de pruebas se ejecutan sin problemas.

Compila el paquete:

```
make
```

La compilación está completa. Como se mencionó antes, no es obligatorio ejecutar los bancos de pruebas de las herramientas temporales en este capítulo. Si de todas formas deseas ejecutar el banco de pruebas de Glibc, hazlo con el siguiente comando:

```
make check
```

Consulta en la Sección 6.9, “Glibc-2.6.1”, la explicación de los fallos de las pruebas que tienen una particular importancia.

En este capítulo algunas pruebas pueden verse afectadas adversamente por las herramientas existentes o el entorno del sistema anfitrión. En resumen, no te preocupes demasiado si ves fallos en el banco de pruebas de Glibc en este capítulo. La Glibc del Capítulo 6 es la que acabaremos usando al final, por lo que es la que necesitamos que pase la mayoría de las pruebas (incluso en el Capítulo 6 es posible que todavía ocurran algunos fallos, la prueba *math* por ejemplo).

Cuando aparezca un fallo, anótalo y continua ejecutando de nuevo **make check**. El banco de pruebas debería continuar a partir de donde se quedó. Puedes evitar esta secuencia de inicio-parada ejecutando **make -k check**. Si utilizas esta opción, asegúrate de registrar la salida para que más tarde puedas revisar el fichero de registro en búsqueda de errores.

La fase de instalación de Glibc mostrará un aviso inofensivo sobre la ausencia del fichero `/tools/etc/ld.so.conf`. Evita este confuso aviso con:

```
mkdir -v /tools/etc
touch /tools/etc/ld.so.conf
```

Instala el paquete:

```
make install
```

Diferentes países y culturas tienen diferentes convenciones sobre cómo comunicarse. Estas convenciones van desde las más simples, como el formato para representar fechas y horas, a las más complejas, como el lenguaje hablado. La "internacionalización" de los programas GNU funciona mediante el uso de *locales*.

Nota

Si no estás ejecutando los bancos de pruebas en este capítulo, como recomendamos, no hay razón para instalar ahora las locales. Las instalaremos en el siguiente capítulo. Si de todas formas deseas instalar las locales, usa las instrucciones que se encuentran en Sección 6.9, "Glibc-2.6.1."

Los detalles sobre este paquete se encuentran en la Sección 6.9.4, "Contenido de Glibc".

5.7. Ajustar las herramientas

Ahora que se han instalado las librerías de C temporales, todas las herramientas que compilemos en el resto de este capítulo deberían enlazarse contra ellas. Para conseguirlo, deben ajustarse el enlazador y el fichero specs del compilador.

El enlazador, que se ajustó al final del primer paso de Binutils, debe renombrarse para que pueda ser encontrado y utilizado correctamente. Primero, haz una copia de seguridad del enlazador original, despues reemplazalo con el enlazador ajustado. También haremos un enlace a su contraparte en `/tools/$(gcc -dumpmachine)/bin`:

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

Desde ahora todo se enlazará solamente contra las librerías que hay en `/tools/lib`.

Lo siguiente esapuntar GCC al nuevo enlazador dinámico. Esto se hace volcando el fichero “specs”de GCC a un lugar en el que GCC lo busque por defecto. Una simple sustitución **sed** cambia el enlazador dinámico que será usado por GCC.

Asegúrate de revisar visualmente el fichero specs para verificar que cada aparición de “/lib/ld-linux.so.2” ha sido reemplazada por “/tools/lib/ld-linux.so.2”:

Importante

Si estás trabajando sobre una plataforma en la que el nombre del enlazador dinámico no es `ld-linux.so.2`, en el siguiente comando debes sustituir `ld-linux.so.2` con el nombre del enlazador dinámico de tu plataforma. En caso necesario consulta la Sección 5.2, “Notas técnicas sobre las herramientas”.

```
gcc -dumpspecs | sed 's@/lib/ld-linux.so.2@/tools&@g' \
  > `dirname $(gcc -print-libgcc-file-name)`/specs
```

Durante el proceso de construcción, GCC ejecuta un guión (**fixincludes**) que explora el sistema buscando ficheros de cabecera que puedan necesitar ser corregidos (que pueden contener errores de sintaxis, por ejemplo), e instala las versiones corregidas en un directorio privado. Existe la posibilidad de que, como resultado de este proceso, algunos ficheros de cabecera del sistema anfitrión se hayan colado dentro de dicho directorio privado de cabeceras de GCC. Como el resto de este capítulo sólo necesita las cabeceras de GCC y Glibc, que ya han sido instaladas, cualquier cabecera “fijada” puede borrarse sin problemas. Esto ayuda a evitar que cualquier cabecera del anfitrión contamine el entorno de construcción. Ejecuta los siguientes comandos para eliminr dichos ficheros de cabecera (puede que encuentres más facil copiar y pegar estos comandos en vez de teclearlos, debido a su longitud):

```
GCC_INCLUDEDIR=`dirname $(gcc -print-libgcc-file-name)`/include &&
find ${GCC_INCLUDEDIR}/* -maxdepth 0 -xtype d -exec rm -rvf '{}' \; &&
rm -vf `grep -l "DO NOT EDIT THIS FILE" ${GCC_INCLUDEDIR}/*` &&
unset GCC_INCLUDEDIR
```

Atención

En este punto es obligatorio parar y asegurarse de que las operaciones básicas (compilación y enlazado) de las nuevas herramientas funcionan como se espera. Para esto vamos a hacer una simple comprobación:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

[Intérprete de programa solicitado:
    /tools/lib/ld-linux.so.2]
```

Confirma que `/tools/lib` aparezca como el prefijo de tu enlazador dinámico.

Si no recibes una salida como la mostrada arriba, o no hay salida alguna, algo está seriamente mal. Investiga y revisa tus pasos para encontrar el problema y corregirlo. El problema debe resolverse antes de continuar. Primero, repite la comprobación de sanidad usando **gcc** en vez de **cc**. Si esto funciona significa que falta el enlace simbólico `/tools/bin/cc`. Vuelve a la Sección 5.4, “GCC-4.2.1 - Fase 1” e instala el enlace simbólico. Seguidamente, asegúrate de que tu `PATH` es correcto. Puedes comprobarlo ejecutando **echo $PATH** y verificando que `/tools/bin` está en cabeza de la lista. Si el `PATH` está mal puede significar que no has ingresado como usuario `lfs` o que algo salió mal en la Sección 4.4, “Configuración del entorno”. Otra opción es que algo pudo ir mal en el anterior arreglo del fichero specs. En este caso, repite el arreglo del fichero.

Cuando todo esté bien, borra los ficheros de prueba:

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

Nota

La construcción de TCL en la siguiente sección servirá como comprobación adicional de que las herramientas se han construido correctamente. Si la construcción de TCL falla, esto es una indicación de que algo fué mal durante la instalación de Binutils, GCC o Glibc, pero no con el propio TCL.

5.8. Tcl-8.4.15

El paquete Tcl contiene el Tool Command Language (Lenguaje para Herramientas de Comandos).

Tiempo estimado de construcción: 0.3 SBU
Espacio requerido en disco: 24 MB

5.8.1. Instalación de Tcl

Este paquete y los dos siguientes (Expect y DejaGNU) se instalan con el único propósito de poder ejecutar los bancos de pruebas de GCC y Binutils. Instalar tres paquetes sólo para realizar comprobaciones puede parecer excesivo, pero es muy tranquilizador, si no esencial, saber que las herramientas más importantes funcionan adecuadamente. Aunque los bancos de pruebas no se ejecuten en este capítulo (no son obligatorios), estos paquetes son todavía necesarios para los bancos de pruebas en el Capítulo 6.

Prepara Tcl para su compilación:

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

Construye el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`TZ=UTC make test`**. Se sabe que el banco de pruebas de Tcl experimenta fallos bajo ciertas condiciones del anfitrión que aún no se comprenden por completo. Sin embargo, estos fallos no son una sorpresa y no se consideran críticos. El parámetro *`TZ=UTC`* establece la zona horaria al Tiempo Universal Coordinado (UTC), también conocido como Hora del Meridiano de Greenwich (GMT), pero sólo mientras se ejecuta el banco de pruebas. Esto asegura que las pruebas de reloj se ejecuten correctamente. En el Capítulo 7 se proporcionan detalles sobre la variable de entorno `TZ`.

Instala el paquete:

```
make install
```

Instala las cabeceras de Tcl. El siguiente paquete, Expect, las necesita para construirse.

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

Crea un enlace simbólico necesario:

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

5.8.2. Contenido de Tcl

Programas instalados: tclsh (enlace a tclsh8.4) y tclsh8.4
Librería instalada: libtcl8.4.so

Descripciones cortas

tclsh8.4 Es el intérprete de comandos de Tcl.

tclsh	Enlace a tclsh8.4
`libtcl8.4.so`	La librería Tcl.

5.9. Expect-5.43.0

El paquete Expect suministra un programa que mantiene diálogos programados con otros programas interactivos.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 4 MB

5.9.1. Instalación de Expect

Corrige un error que puede causar falsos fallos durante la ejecución del banco de pruebas de GCC:

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

Fuerza al guión de configuración de Expect a usar `/bin/stty` en vez de un posible `/usr/local/bin/stty` que podría encontrarse en el sistema anfitrión. Esto asegurará que nuestras herramientas de pruebas permanezcan sanas para la contrucción final de las herramientas principales:

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

Prepara Expect para su compilación:

```
./configure --prefix=/tools --with-tcl=/tools/lib \
  --with-tclinclude=/tools/include --with-x=no
```

Significado de las opciones de configure:

`--with-tcl=/tools/lib`
Esto asegura que el guión configure encuentre la instalación de Tcl en nuestra ubicación temporal de herramientas. No queremos que encuentre una que pudiese residir en el sistema anfitrión.

`--with-tclinclude=/tools/include`
Esto le especifica a Expect dónde encontrar las cabeceras internas de Tcl. El uso de esta opción evita los casos en que **configure** falla debido a que no puede encontrar automáticamente la localización de las cabeceras de Tcl.

`--with-x=no`
Esto le indica al guión configure que no busque Tk (el componente GUI de Tcl) o las librerías del sistema X Window, las cuales posiblemente se encuentren en el sistema anfitrión pero no existirán en el entorno temporal.

Construye el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make test`**. Sin embargo, se sabe que el banco de pruebas para Expect a veces experimenta fallos bajo ciertas condiciones del anfitrión que no están bajo nuestro control. Por tanto, estos fallos del banco de pruebas no son una sorpresa y no se consideran críticos.

Instala el paquete:

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

Significado del parámetro de make:

`SCRIPTS=""`
Esto evita la instalación de los guiones suplementarios de Expect, que no son necesarios.

5.9.2. Contenido de Expect

Programa instalado: expect
Librería instalada: libexpect-5.43.a

Descripciones cortas

expect Se comunica con otros programas interactivos según un guión.

`libexpect-5.43.a` Contiene funciones que permiten a Expect usarse como una extensión de Tcl o usarse directamente en C o C++ (sin Tcl)."

5.10. DejaGNU-1.4.4

El paquete DejaGNU contiene un entorno de trabajo para comprobar otros programas.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 6.2 MB

5.10.1. Instalación de DejaGNU

Prepara DejaGNU para su compilación:

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

Construye e instala el paquete:

```
make install
```

Para comprobar los resultados, ejecuta: **`make check`**.

5.10.2. Contenido de DejaGNU

Programa instalado: runtest

Descripción corta

runtest Un guión envoltorio que encuentra el intérprete de comandos de **expect** adecuado y entonces ejecuta DejaGNU.

5.11. GCC-4.2.1 - Fase 2

El paquete GCC contiene la colección de compiladores GNU, que incluye los compiladores C y C++.

Tiempo estimado de construcción:	4.2 SBU
Espacio requerido en disco:	553 MB

5.11.1. Reinstalación de GCC

Ahora están instaladas las herramientas necesarias para comprobar GCC y Binutils: Tcl, Expect y DejaGNU. Por lo que ahora pueden reconstruirse GCC y Binutils enlazándolos con la nueva Glibc y comprobarlos adecuadamente (si llevas a cabo los bancos de pruebas en este capítulo). Sin embargo, una cosa a tener en cuenta es que estos bancos de pruebas son altamente dependientes del correcto funcionamiento de las pseudo-terminales (PTYs) suministradas por tu distribución anfitrión. Las PTYs se implementan normalmente mediante el sistema de ficheros `devpts`. Comprueba si tu sistema anfitrión está configurado correctamente en este aspecto ejecutando una simple prueba:

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

La respuesta podría ser:

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

El sistema no tiene más ptys.
Pídele al administrador del sistema que cree más.
```

Si recibes el mensaje anterior, tu sistema anfitrión no está configurado para operar correctamente con PTYs. En este caso no hay razón para ejecutar los bancos de pruebas de GCC y Binutils hasta resolver este asunto. Puedes consultar la FAQ de LFS en *http://www.linuxfromscratch.org/faq//lfs/faq.html#no-ptys* para obtener información sobre cómo conseguir que funcionen las PTYs.

Como se explicó anteriormente en Sección 5.7, “Ajustar las herramientas”, bajo circunstancias normales el guión **fixincludes** de GCC se ejecuta para corregir posibles ficheros de cabecera rotos. Como GCC-4.2.1 y Glibc-2.6.1 ya han sido instalados en este punto, y se sabe que sus respectivos ficheros de cabecera no necesitan ser corregidos, no se necesita el guión **fixincludes**. Como se mencionó anteriormente, de hecho el guión puede contaminar el entorno de construcción al instalar cabeceras corregidas procedentes del anfitrión dentro del directorio privado de cabeceras de GCC. La ejecución del guión **fixincludes** puede evitarse ejecutando los siguientes comandos:

```
cp -v gcc/Makefile.in{,.orig}
sed 's@\./fixinc\.sh@-c true@' gcc/Makefile.in.orig > gcc/Makefile.in
```

El proceso "bootstrap" realizado en Sección 5.4, “GCC-4.2.1 - Fase 1” construye GCC con la opción `-fomit-frame-pointer`. Las construcciones no "bootstrap" omiten dicha opción, así que aplica el siguiente **sed** para usarla y asegurar construcciones consistentes del compliador:

```
cp -v gcc/Makefile.in{,.tmp}
sed 's/^XCFLAGS =$/& -fomit-frame-pointer/' gcc/Makefile.in.tmp \
  > gcc/Makefile.in
```

El siguiente comando cambiará la localización del enlazador dinámico usado por GCC para usar el instalado en `/tools`. También elimina `/usr/include` de la ruta de búsqueda de inclusiones de GCC. Al hacer esto ahora en vez de ajustar el fichero de especificaciones tras la instalación se asegura que el nuevo enlazador dinámico se utiliza durante la construcción actual de GCC. Esto es, todos los binarios creados durante la construcción se enlazarán contra la nueva Glibc. Ejecuta:

```
for file in $(find gcc/config -name linux64.h -o -name linux.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_INCLUDE_DIR
#define STANDARD_INCLUDE_DIR 0" >> $file
  touch $file.orig
done
```

En caso de que lo anterior parezca dificil de entender, vamos a analizarlo. Primero se encuentran todos los ficheros bajo el directorio gcc/config cuyo nombre sea `linux.h` o `linux64.h`. Cada fichero encontrado es copiado a un fichero del mismo nombre pero añadiendo el sufijo ".orig". Entonces la primera expresión sed añade "/tools" delante de cada aparición de "/lib/ld", "/lib64/ld" o "/lib32/ld", mientras que la segunda reemplaza "/usr". Entonces se añaden al final del fichero las sentencias de definición que modifican las rútas de búsqueda de inclusiones. Por último, se utiliza **touch** para actualizar la marca de tiempo de los ficheros copiados. Al usarlo junto con **cp -u** se evitan cambios no deseados en los ficheros originales en caso de que el comando se ejecute dos veces inadvertidamente.

Vuelve a crear un directorio de construcción dedicado:

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

Prepara GCC para su compilación:

```
../gcc-4.2.1/configure --prefix=/tools \
    --with-local-prefix=/tools --enable-clocale=gnu \
    --enable-shared --enable-threads=posix \
    --enable-__cxa_atexit --enable-languages=c,c++ \
    --disable-libstdcxx-pch --disable-bootstrap
```

Significado de las nuevas opciones de configure:

`--enable-clocale=gnu`
Esta opción asegura que se seleccione el modelo de locale correcto para las librerías de C++ en todos los casos. Si el guión configure encuentra instalada la locale *de_DE*, seleccionará el modelo correcto de *gnu*. Sin embargo, las personas que no instalan la locale *de_DE* pueden correr el riesgo de construir librerías de C++ incompatibles en la ABI debido a que se selecciona el modelo de locale genérico, que es incorrecto.

`--enable-threads=posix`
Esto activa el manejo de excepciones C++ para código multihilo.

`--enable-__cxa_atexit`
Esta opción permite el uso de `__cxa_atexit`, en vez de `atexit`, para registrar destructores C++ para objetos estáticos locales y objetos globales. Es esencial para un manejo de destructores completamente compatible con

los estándares. También afecta al ABI de C++ obteniendo librerías compartidas y programas C++ interoperables con otras distribuciones Linux.

`--enable-languages=c,c++`

Esta opción asegura que se construyan tanto el compilador de C como el de C++.

`--disable-libstdcxx-pch`

No construye la cabecera precompilada (PCH) para `libstdc++`. Necesita mucho espacio y nosotros no la utilizamos.

`--disable-bootstrap`

Ahora GCC se auto-reconstruye (bootstrapping) por defecto. Sin embargo, nuestro método de construcción debería proporcionarnos un compilador sólido sin necesidad de auto-reconstruirlo cada vez.

Compila el paquete:

```
make
```

Aquí no hace falta usar el objetivo *`bootstrap`*, ya que el compilador que estamos utilizando para construir GCC ha sido construido a partir de la misma versión de las fuentes de GCC que usamos antes.

La compilación está completa. Como se mencionó antes, no es obligatorio ejecutar los bancos de pruebas de las herramientas temporales en este capítulo. Si de todas formas deseas ejecutar el banco de pruebas de GCC, hazlo con el siguiente comando:

```
make -k check
```

La opción *`-k`* se usa para que el banco de pruebas se ejecute por completo y sin detenerse ante el primer error. El banco de pruebas de GCC es muy exhaustivo y es casi seguro que generará algunos fallos.

Para una relación de las pruebas fallidas más importantes, mira en Sección 6.12, “GCC-4.2.1.”

Instala el paquete:

```
make install
```

Atención

En este punto es obligatorio parar y asegurarse de que las operaciones básicas (compilación y enlazado) de las nuevas herramientas funcionan como se espera. Para esto vamos a hacer una simple comprobación:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

[Intérprete de programa solicitado:
    /tools/lib/ld-linux.so.2]
```

Confirma que `/tools/lib` aparezca como el prefijo de tu enlazador dinámico.

Si no recibes una salida como la mostrada arriba, o no hay salida alguna, algo está seriamente mal. Investiga y revisa tus pasos para encontrar el problema y corregirlo. El problema debe resolverse antes de continuar. Primero, repite la comprobación de sanidad usando **gcc** en vez de **cc**. Si esto funciona significa que falta el enlace simbólico `/tools/bin/cc`. Vuelve a la Sección 5.4, “GCC-4.2.1 - Fase 1” e instala el enlace simbólico. Seguidamente, asegúrate de que tu `PATH` es correcto. Puedes comprobarlo ejecutando **echo $PATH** y verificando que `/tools/bin` está en cabeza de la lista. Si el `PATH` está mal puede significar que no has ingresado como usuario `lfs` o que algo salió mal en la Sección 4.4, “Configuración del entorno”. Otra opción es que algo pudo ir mal en el anterior arreglo del fichero specs. En este caso, repite el arreglo del fichero.

Cuando todo esté bien, borra los ficheros de prueba:

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

Los detalles sobre este paquete se encuentran en la Sección 6.12.2, “Contenido de GCC”.

5.12. Binutils-2.18 - Fase 2

El paquete Binutils contiene un enlazador, un ensamblador y otras utilidades para trabajar con ficheros objeto.

Tiempo estimado de construcción:	1 SBU
Espacio requerido en disco:	177 MB

5.12.1. Reinstalación de Binutils

Vuelve a crear un directorio dedicado para la construcción:

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

Prepara Binutils para su compilación:

```
../binutils-2.18/configure --prefix=/tools \
    --disable-nls --with-lib-path=/tools/lib
```

Significado de la nueva opción de configure:

`--with-lib-path=/tools/lib`
Esto le indica al guión configure que especifique la ruta de búsqueda de librerías por defecto durante la compilación de Binutils, resultando en que se le pase `/tools/lib` al enlazador. Esto evita que el enlazador busque en los directorios de librerías del anfitrión.

Compila el paquete:

```
make
```

La compilación está completa. Como se explicó antes, no recomendamos ejecutar los bancos de pruebas de las herramientas temporales en este capítulo. Si de todas formas deseas ejecutar el banco de pruebas de Binutils, hazlo con el siguiente comando:

```
make check
```

Instala el paquete:

```
make install
```

Prepara el enlazador para la fase de “Reajuste” del próximo capítulo:

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

Los detalles sobre este paquete se encuentran en la Sección 6.11.2, “Contenido de Binutils”.

5.13. Ncurses-5.6

El paquete Ncurses contiene librerías para el manejo de pantallas de caracteres independiente del terminal.

Tiempo estimado de construcción: 0.7 SBU
Espacio requerido en disco: 30 MB

5.13.1. Instalación de Ncurses

Prepara Ncurses para su compilación:

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

Significado de las opciones de configure:

`--without-ada`
Esto asegura que Ncurses no construya su soporte para el compilador Ada, que puede estar presente en el anfitrión pero que no estará disponible al entrar en el entorno **chroot**.

`--enable-overwrite`
Esto le indica a Ncurses que instale sus ficheros de cabecera en `/tools/include` en vez de en `/tools/include/ncurses` para asegurar que otros paquetes puedan encontrar sin problemas las cabeceras de Ncurses.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.20.2, “Contenido de Ncurses”.

5.14. Bash-3.2

El paquete Bash contiene la “Bourne-Again SHell”.

Tiempo estimado de construcción:	0.4 SBU
Espacio requerido en disco:	22 MB

5.14.1. Instalación de Bash

Aplica correcciones para varios fallos descubiertos dede la publicación inicial de Bash-3.2:

```
patch -Np1 -i ../bash-3.2-fixes-6.patch
```

Prepara Bash para su compilación:

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

Significado de la opción de configure:

`--without-bash-malloc`
Esta opción desactiva el uso de la función de ubicación de memoria (`malloc`) de Bash, que se sabe que provoca violaciones de segmento. Al desactivar esta opción Bash utilizará la función `malloc` de Glibc, que es más estable.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make tests`**.

Instala el paquete:

```
make install
```

Crea un enlace para los programas que usan **sh** como intérprete de comandos:

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

Los detalles sobre este paquete se encuentran en la Sección 6.28.2, “Contenido de Bash”.

5.15. Bzip2-1.0.4

El paquete Bzip2 contiene programas para comprimir y descomprimir ficheros. Comprimir ficheros de texto con **bzip2** proporciona un mayor porcentaje de compresión que el tradicional **gzip**.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	3.5 MB

5.15.1. Instalación de Bzip2

El paquete Bzip2 no tiene un guión **configure**. Compílalo y comprueba los resultados con:

```
make
```

Instala el paquete:

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

Los detalles sobre este paquete se encuentran en la Sección 6.29.2, “Contenido de Bzip2”.

5.16. Coreutils-6.9

El paquete Coreutils contiene utilidades para mostrar y establecer las características básicas del sistema.

Tiempo estimado de construcción: 0.5 SBU
Espacio requerido en disco: 67.6 MB

5.16.1. Instalación de Coreutils

La versión de la función “futimens” usada por Coreutils es incompatible con la versión actual proporcionada por Glibc. Por tanto, renombra la función:

```
for file in src/{copy,touch}.c lib/utimens.{c,h} ; do \
   cp -v $file{,.orig}
   sed 's/futimens/gl_&/' $file.orig > $file
done
```

Prepara Coreutils para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make RUN_EXPENSIVE_TESTS=yes check`**. El parámetro *`RUN_EXPENSIVE_TESTS=yes`* le indica al banco de pruebas que realice varias comprobaciones adicionales que se consideran relativamente costosas (en términos de uso de CPU y memoria) en ciertas plataformas, aunque normalmente no hay problemas en Linux.

Instala el paquete:

```
make install
```

El comando anterior no instala `su` debido a que no puede instalarlo con setuid a root desde un usuario sin privilegios. Instalandolo manualmente con un nombre diferente podremos ejecutar ciertos bancos de pruebas del sistema final como usuario sin privilegios mientras mantenemos el posiblemente util **su** de nuestro anfitrión en nuestro PATH. Instalalo con:

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

Los detalles sobre este paquete se encuentran en la Sección 6.16.2, “Contenido de Coreutils”.

5.17. Diffutils-2.8.1

El paquete Diffutils contiene programas que muestran las diferencias entre ficheros o directorios.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	6.2 MB

5.17.1. Instalación de Diffutils

Prepara Diffutils para su compilación:

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

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.30.2, “Contenido de Diffutils”.

5.18. Findutils-4.2.31

El paquete Findutils contiene programas para encontrar ficheros. Se suministran estos programas para hacer búsquedas recursivas en un árbol de directorios, y para crear, mantener y consultar una base de datos (más rápida que la búsqueda recursiva, pero imprecisa si la base de datos no se ha actualizado recientemente).

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 13.6 MB

5.18.1. Instalación de Findutils

Prepara Findutils para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.32.2, “Contenido de Findutils”.

5.19. Gawk-3.1.5

El paquete Gawk contiene programas para manipular ficheros de texto.

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 18.2 MB

5.19.1. Instalación de Gawk

Prepara Gawk para su compilación:

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

Debido a un fallo en el guión **configure**, Gawk falla al detectar ciertos aspectos del soporte para locales de Glibc. Este error provoca, por ejemplo, fallos en el banco de pruebas de Gettext. Evita este problema añadiendo las definiciones de macro ausentes en `config.h`:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.35.2, “Contenido de Gawk”.

5.20. Gettext-0.16.1

El paquete Gettext contiene utilidades para la internacionalización y localización. Esto permite a los programas compilarse con Soporte de Lenguaje Nativo (NLS), lo que les permite mostrar mensajes en el idioma nativo del usuario.

Tiempo estimado de construcción: 0.4 SBU
Espacio requerido en disco: 43 MB

5.20.1. Instalación de Gettext

Para nuestro conjunto de harramientas temporales sólo necesitamos compilar e instalar un programa de Gettext.

Prepara Gettext para su compilación:

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

Significado de la opción de configure:

`--disable-shared`
No necesitamos instalar por ahora ninguna de las librerías compartidas de Gettext, por tanto no necesitamos construirlas.

Compila el paquete:

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

Como sólo se ha compilado un binario, no es posible ejecutar el banco de pruebas sin compilar librerías de soporte adicionales del paquete Gettext. Por tanto no se recomienda intentar ejecutar el banco de pruebas en esta fase.

Instala el binario **msgfmt**:

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

Los detalles sobre este paquete se encuentran en la Sección 6.36.2, “Contenido de Gettext”.

5.21. Grep-2.5.1a

El paquete Grep contiene programas para buscar dentro de ficheros.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	4.8 MB

5.21.1. Instalación de Grep

Prepara Grep para su compilación:

```
./configure --prefix=/tools \
    --disable-perl-regexp
```

Significado de la opción de configure:

`--disable-perl-regexp`
Esto asegura que **grep** no se enlaza contra alguna librería PCRE que pudiese estar presente en el anfitrión y que no estará disponible una vez que entremos en el entorno **chroot**.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.37.2, “Contenido de Grep”.

5.22. Gzip-1.3.12

El paquete Gzip contiene programas para comprimir y descomprimir ficheros.

Tiempo estimado de construcción:	less than 0.1 SBU
Espacio requerido en disco:	2.2 MB

5.22.1. Instalación de Gzip

La versión de la función “futimens” usada por Gzip es incompatible con la versión actual proporcionada por Glibc. Por tanto, renombra la función:

```
for file in gzip.c lib/utimens.{c,h} ; do \
   cp -v $file{,.orig}
   sed 's/futimens/gl_&/' $file.orig > $file
done
```

Prepara Gzip para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.39.2, “Contenido de Gzip”.

5.23. Make-3.81

El paquete Make contiene un programa para compilar paquetes.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	9.6 MB

5.23.1. Instalación de Make

Prepara Make para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.44.2, “Contenido de Make”.

5.24. Patch-2.5.4

El paquete Patch contiene un programa para modificar o crear ficheros mediante la aplicación de un fichero “parche” creado normalmente con el programa **diff**.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 1.6 MB

5.24.1. Instalación de Patch

Prepara Patch para su compilación:

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

Compila el paquete:

```
make
```

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.48.2, “Contenido de Patch”.

5.25. Perl-5.8.8

El paquete Perl contiene el Lenguaje Práctico de Extracción e Informe.

Tiempo estimado de construcción:	0.7 SBU
Espacio requerido en disco:	84 MB

5.25.1. Instalación de Perl

Aplica el siguiente parche para corregir algunas rutas a la librería C fijadas en el código:

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

Corrige una incompatibilidad con gcc-4.2.1:

```
mv -v makedepend.SH{,.orig}
sed 's/command /command[ -]/' makedepend.SH.orig > makedepend.SH
```

Prepara Perl para su compilación (asegúrate de poner correctamente 'Data/Dumper Fcntl IO POSIX', todo son letras):

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

Significado de la opción de configure:

`-Dstatic_ext='Data/Dumper Fcntl IO POSIX'`
Esto le indica a Perl que construya el conjunto mínimo de extensiones estáticas necesarias para ejecutar el banco de pruebas de Coreutils y Glibc en el siguiente capítulo.

Sólo es necesario construir algunas de las utilidades incluidas en este paquete:

```
make perl utilities
```

Aunque Perl incluye un banco de pruebas, no es recomendable ejecutarlo ahora. Sólo se ha construido una parte de Perl y la ejecución de **make test** provocaría que también se compilase el resto de Perl, que es innecesario en este momento. El banco de pruebas puede ejecutarse en el siguiente capítulo, si se desea.

Instala estas herramientas y sus librerías:

```
cp -v perl pod/pod2man /tools/bin
mkdir -pv /tools/lib/perl5/5.8.8
cp -Rv lib/* /tools/lib/perl5/5.8.8
```

Los detalles sobre este paquete se encuentran en la Sección 6.23.2, “Contenido de Perl”.

5.26. Sed-4.1.5

El paquete Sed contiene un editor de flujos.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	6.1 MB

5.26.1. Instalación de Sed

Prepara Sed para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.14.2, “Contenido de Sed”.

5.27. Tar-1.18

El paquete Tar contiene un programa de archivado.

Tiempo estimado de construcción:	0.3 SBU
Espacio requerido en disco:	19.9 MB

5.27.1. Instalación de Tar

Prepara Tar para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.53.2, “Contenido de Tar”.

5.28. Texinfo-4.9

El paquete Texinfo contiene programas usados para leer, escribir y convertir páginas info.

Tiempo estimado de construcción:	0.2 SBU
Espacio requerido en disco:	16.3 MB

5.28.1. Instalación de Texinfo

Prepara Texinfo para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 6.54.2, “Contenido de Texinfo”.

5.29. Util-linux-2.12r

El paquete Util-linux contiene una miscelánea de utilidades. Entre otras hay utilidades para manejar sistemas de ficheros, consolas, particiones y mensajes.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 8.9 MB

5.29.1. Instalación de Util-linux

Util-linux no utiliza las cabeceras y librerías recién instaladas en el directorio `/tools`. Esto se corrige modificando el guión configure:

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

Prepara Util-linux para su compilación:

```
./configure
```

Construye algunas rutinas de soporte:

```
make -C lib
```

Sólo es necesario construir algunas de las utilidades incluidas en este paquete:

```
make -C mount mount umount
make -C text-utils more
```

Este paquete no incluye un banco de pruebas.

Copia estos programas al directorio de herramientas temporales:

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

Los detalles sobre este paquete se encuentran en la Sección 6.56.3, “Contenido de Util-linux”.

5.30. Eliminación de Símbolos

Los pasos de esta sección son opcionales, pero si la partición LFS es pequeña es bueno saber que se pueden eliminar algunas cosas innecesarias. Los binarios y librerías que se han construido contienen unos 70 MB de símbolos de depuración innecesarios. Elimina esos símbolos con:

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

El último de los comandos anteriores se saltará una veintena de ficheros, avisando que no reconoce su formato. Muchos de ellos son guiones en vez de binarios.

Ten cuidado de *no* utilizar *--strip-unneeded* con las librerías. Las estáticas se destruirían y tendrías que construir de nuevo los tres paquetes de las herramientas principales.

Para recuperar unos 20 MB mas, elimina la documentación:

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

En este momento deberías tener como mínimo 850 MB de espacio libre en `$LFS` para poder construir e instalar Glibc en el siguiente capítulo. Si puedes construir e instalar Glibc, podrás construir e instalar el resto.

5.31. Cambio del propietario

Nota

Tanto estos comandos como los del resto del libro deben realizarse como usuario `root`, no como usuario `lfs`. Igualmente, vuelve a comprobar que `$LFS` está definido en el entorno de `root`.

En estos momentos el directorio `$LFS/tools` pertenece al usuario `lfs`, que sólo existe en el sistema anfitrión. Si el directorio `$LFS/tools` se conserva como está, los ficheros pertenecerán a un ID de usuario sin su correspondiente cuenta. Esto es peligroso porque una cuenta de usuario creada posteriormente podría tener este ID de usuario y podría poseer el directorio `$LFS/tools` y todos los ficheros que contiene, exponiéndolos a una posible manipulación maliciosa.

Para evitar este problema, puedes añadir el usuario `lfs` al nuevo sistema LFS cuando creemos el fichero `/etc/passwd`, teniendo cuidado de asignarle los mismos identificadores de usuario y grupo que en el sistema anfitrión. Mejor aún, cambia el propietario del directorio `$LFS/tools` al usuario `root` ejecutando el siguiente comando:

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

Aunque el directorio `$LFS/tools` puede ser borrado una vez terminado el sistema LFS, puede ser guardado para construir sistemas LFS adicionales *de esta misma versión del libro*. Cual es la mejor forma de guardar el directorio `$LFS/tools` es una cuestión de preferencias, y se deja como ejercicio para el lector.

Atención

Si piensas guardar las herramientas temporales para construir otros sistemas LFS en el futuro, *ahora* es el momento de hacerlo. Los siguientes comandos realizados en el capítulo 6 modificarán las herramientas temporales, haciendolas inservibles para construcciones futuras.

Parte III. Construcción del sistema LFS

Capítulo 6. Instalación de los programas del sistema base

6.1. Introducción

En este capítulo entramos en la zona de edificación y comenzamos a construir de verdad nuestro sistema LFS. Es decir, cambiamos la raíz a nuestro mini sistema Linux temporal, hacemos unos cuantos preparativos finales, y entonces comenzamos a instalar los paquetes.

La instalación de estos programas es bastante sencilla. Aunque en muchos casos las instrucciones podrían acortarse y ser más genéricas, hemos optado por suministrar las instrucciones completas para cada paquete para minimizar la posibilidad de errores. La clave para aprender qué hace que un sistema Linux funcione es conocer para qué se utiliza cada paquete y por qué el usuario (o el sistema) lo necesita. Para cada paquete instalado se incluye un sumario con su contenido, seguido de una concisa descripción de cada programa y librería instalados por el paquete.

Si piensas usar optimizaciones para la compilación durante este capítulo, mírate la receta de optimización en *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*. Las optimizaciones del compilador pueden hacer que un programa funcione más rápido, pero también pueden dificultar la compilación e incluso dar problemas al ejecutar el programa. Si un paquete rehusa compilar cuando se usan optimizaciones, prueba a compilarlo sin ellas y mira si el problema desaparece. Incluso si el paquete se compila usando optimización, existe el riesgo de que pueda haberse compilado incorrectamente debido a las complejas interacciones entre el código y las herramientas de construcción. Ten en cuenta también que las opciones `-march` y `-mtune` pueden causar problemas en las herramientas principales (Binutils, GCC y Glibc). La pequeña ganancia que se consigue usando optimizaciones en la compilación generalmente queda ensombrecida por los riesgos. Aconsejamos a los constructores primerizos de LFS que construyan sin optimizaciones personalizadas. Tu sistema aún será muy rápido y, al mismo tiempo, muy estable.

El orden en el que se instalan los paquetes en este capítulo debe respetarse estrictamente para asegurar que ningún programa inserte en su código una ruta referente a `/tools`. Por la misma razón, no compiles paquetes en paralelo. La compilación en paralelo puede ahorrarte algo de tiempo (sobre todo en máquinas con CPUs duales), pero puede generar un programa que contenga referencias a `/tools`, lo que provocaría que el programa dejase de funcionar cuando se elimine dicho directorio.

Antes de las instrucciones de instalación de cada paquete se muestra algo de información sobre el mismo: una breve descripción de lo que contiene, cuánto tardará aproximadamente en construirse y cuánto espacio en disco necesita durante el proceso de construcción. A las instrucciones de instalación le sigue una lista de los programas y librerías que instala el paquete, junto con sus descripciones cortas.

6.2. Preparar los sistemas de ficheros virtuales del núcleo

Varios sistemas de ficheros exportados por el núcleo son usados para comunicarse hacia y desde el propio núcleo. Estos sistemas de ficheros son virtuales y no utilizan espacio en disco. El contenido del sistema de ficheros reside en memoria.

Comienza creando los directorios sobre los que se montarán dichos sistemas de ficheros:

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

6.2.1. Crear los nodos de dispositivo iniciales

Cuendo el núcleo arranca el sistema, este necesita la presencia de ciertos nodos de dispositivo, en particular los dispositivos `console` y `null`. Los nodos de dispositivo serán creados en el disco duro para que estén disponibles antes de que **udevd** sea iniciado, y adicionalmente cuando Linux es iniciado con *`init=/bin/bash`*. Crea los dispositivos ejecutando los siguientes comandos:

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

6.2.2. Montar y poblar /dev

El método recomendado para poblar el directorio `/dev` con dispositivos es montar un sistema de ficheros virtual (como `tmpfs`) en el directorio `/dev`, y permitir que los dispositivos sean creados dinámicamente en dicho sistema de ficheros virtual a medida que son detectados o accedidos. Esto lo hace generalmente Udev durante el arranque. Puesto que este nuevo sistema no tiene aún Udev y no ha sido arrancado, es necesario montar y poblar `/dev` manualmente. Esto se consige mediante un montaje enlazado del directorio `/dev` del sistema anfitrión. Un montaje enlazado es un tipo especial de montaje que te permite crear una replica de un directorio o punto de montaje en otra localización. Utiliza el siguiente comando para hacer esto:

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

6.2.3. Montar los sistemas de ficheros virtuales del núcleo

Ahora monta el resto de sistemas de ficheros virtuales del núcleo:

```
mount -vt devpts devpts $LFS/dev/pts
mount -vt tmpfs shm $LFS/dev/shm
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
```

6.3. Adiministración de paquetes

Frecuentemente se solicita la inclusión de la administración de paquetes en el libro LFS. Un administrador de paquetes permite supervisar la instalación de ficheros facilitando la eliminación y actualización de ficheros. Y antes de que empieces a preguntar, NO, esta sección no habla sobre un administrador de paquetes en concreto, ni recomienda alguno. Lo que suministra es un paseo por las técnicas más populares y su método de trabajo. El administrador de paquetes perfecto para ti puede encontrarse entre estas técnicas o puede ser una combinación de dos o más de ellas. Esta sección menciona brevemente los problemas que pueden surguir cuando se actualizan paquetes.

Algunas razones por las que ningún administrador de paquetes se menciona en LFS or BLFS:

- Tratar la administración de paquetes se sale de los objetivos de estos libros: enseñar cómo se construye un sistema Linux.
- Hay múltiples soluciones para la administración de paquetes, cada una con sus limitaciones y problemática. Incluir uno que satisfaga a todo el mundo es dificil.

Se han escrito diversas recetas sobre este tema. Visita el *Proyecto Hints*para ver si alguna de ellas cubre tus necesidades.

6.3.1. Cuestiones de actualización

Un administrador de paquetes facilita la actualización a nuevas versiones cuando estas son liberadas. Generalmente se pueden usar las instrucciones de los libros LFS y BLFS para actualizar a la nueva versión. A continuación hay algunos puntos que debes tener en cuenta cuando actualices paquetes, especialmente en sistemas en ejecución.

- Si necesitas actualizar uno de los paquetes de las herramientas principales (Glibc, GCC o Binutils) a una nueva versión menor, es más seguro reconstruir el LFS. Aunque *podrías* ser capaz de reconstruir todos los paquetes en su orden de dependencias, no lo recomendamos. Por ejemplo, si necesitas actualizar de glibc-2.2.x a glibc-2.3.x, es más seguro reconstruir. Para actualizaciones de micro-versión, una simple reinstalación funciona normalmente, pero no está garantizado. Por ejemplo, actualizar de glibc-2.3.4 a glibc-2.3.5 no suele causar problemas.
- Si se actualiza un paquete que contiene una librería compartida, y si el nombre de la librería cambia, entonces necesitas recompilar todos los paquetes enlazados dinámicamente a esa librería para que se enlacen contra la nueva. (Advierte que no hay correlación entre la versión del paquete y el nombre de la librería.) Por ejemplo, considera un paquete foo-1.2.3 que instala una librería compartida con el nombre `libfoo.so.1`. Digamos que actualizas el paquete a la nueva versión foo-1.2.4 que instala una librería compartida de nombre `libfoo.so.2`. En este caso, todos los paquetes que están enlazados dinámicamente a `libfoo.so.1` deben recompilarse para que se enlacen contra `libfoo.so.2`. Ten en cuenta que no deberías eliminar las librerías anteriores hasta recompilar los paquetes dependientes.

6.3.2. Técnicas de administración de paquetes

Lo siguiente son algunas técnicas comunes de administración de paquetes. Antes de tomar una decisión sobre un administrador de paquetes, haz una búsqueda de las diversas técnicas, particularmente de los inconvenientes de cada uno.

6.3.2.1. ¡Todos está en mi cabeza!

Si, esta es una técnica de administración de paquetes. Algunas personas no encuentran necesario un administrados de paquetes porque conocen íntimamente los paquetes y saben qué ficheros instala cada paquete. Algunos usuarios tampoco lo necesitan porque piensan reconstruir el sistema al completo cuando cambia un paquete.

6.3.2.2. Instalar en directorios separados

Esta es una administración de paquetes muy simple que no necesita paquetes adicionales para manejar la instalación. Cada paquete se instala en un directorio aparte. Por ejemplo, el paquete foo-1.1 se instala en `/usr/pkg/foo-1.1` y se hace un enlace simbólico de `/usr/pkg/foo` a `/usr/pkg/foo-1.1`. Cuando se instala una nueva versión foo-1.2, esta se instala en `/usr/pkg/foo-1.2` y el anterior enlace se reemplaza por un enlace a la nueva versión.

Las variables de entorno como `PATH`, `LD_LIBRARY_PATH`, `MANPATH`, `INFOPATH` y `CPPFLAGS` deben expandirse para incluir `/usr/pkg/foo`. Para más de unos pocos paquetes este esquema se hace inmanejable.

6.3.2.3. Administración de paquetes por medio de enlaces

Esta es una variante de la técnica anterior. Cada paquete se instala de forma similar a la del esquema anterior. Pero en vez de hacer el enlace, cada fichero se enlaza en la jerarquía `/usr`. Esto elimina la necesidad de ampliar las variables de entorno. Aunque el usuario puede crear los enlaces, para automatizar su creación se han escrito diversos administradores de paquetes basados en este sistema. Algunos de los más populares son Stow, Epkg, Graft, y Depot.

Es necesario falsear la instalación, para que el paquete piense que se instala en `/usr` aunque en realidad sea instalado en la jerarquía `/usr/pkg`. Instalar de esta forma no es una tarea trivial. Por ejemplo, considera que instalas un paquete libfoo-1.1. Las siguientes instrucciones no instalarán el paquete correctamente:

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

La instalación funcionará, pero los paquetes que dependan de ella no se enlazarán con libfoo como cabría esperar. Si compilas un paquete que se enlaza contra libfoo advertirás que se enlaza a `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` en lugar de `/usr/lib/libfoo.so.1` como esperabas. El método correcto es usar la estratégia `DESTDIR` para falsear la instalación del paquete. Este método funciona así:

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

La mayoría de los paquetes soportarán este método, pero algunos no. Con los paquetes que no lo soportan puedes instalarlos manualmente o te será más facil instalar algún paquete problemático en `/opt`.

6.3.2.4. Basado en marcas de fecha

En esta técnica, un fichero es marcado con la fecha antes de instalar el paquete. Tras la instalación, un simple comando **find** con las opciones apropiadas puede generar un registro de todos los ficheros instalados tras la creaciónh del fichero marcado. Un administrador de paquetes escrito con este método es install-log.

Aunque este esquema tiene la ventaja de ser simple, tiene dos inconvenientes. Si durante la instalación los ficheros se instalan con una marca de fecha diferente a la actual, estos ficheros no serán registrados por el administrador de paquetes. Igualmente, este esquema solo puede usarse instalando un paquete cada vez. Los registros no serán válidos si se están instalando dos paquetes desde dos consolas diferentes.

6.3.2.5. Basado en LD_PRELOAD

En este método se precarga una librería antes de la instalación. Durante la instalación esta librería supervisa los paquetes que están siendo instalados adjuntandose ella mismo a varios ejecutables como **cp**, **install**, **mv** y supervisa las llamadas del sistema que modifican el sistema de ficheros. Para que este método funcione todos los ejecutables deben estar enlazados dinámicamente y sin los bits suid o sgid. Precargar la librería puede causar algunos efectos indeseados durante la instalación, por lo que se han de realizar algunas pruebas para asegurar que el administrador de paquetes no rompe nada y registrar todos los ficheros pertinentes.

6.3.2.6. Crear archivos de paquetes

En este esquema la instalación del paquete es falseada dentro de un árbol separado, como se describe en la administración de paquetes por medio de enlaces. Tras la instalación, se crea un archivo del paquete usando los ficheros instalados. Entonces se utiliza este archivo para instalar el paquete en la máquina local, o incluso puede usarse para instalar el paquete en otras máquinas.

Este método es el usado por muchos de los administradores de paquetes que se encuentran en las distribuciones comerciales. Ejemplos de administradores de paquetes que siguen este método son RPM (que es el requerido por *Linux Standard Base Specification*), pkg-utils, apt de Debian y el sistema Portage de Gentoo. Una receta describiendo cómo adaptar este estilo de administración de paquetes a sistemas LFS se encuentra en *http://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt*.

6.3.2.7. Administración basada en usuario

Este esquema, que es propio de LFS, fué desarrollado por Matthias Benkmann y está disponible en el *Proyecto Hints*. En este esquema, cada paquete se instala con un usuario diferente dentro de las localizaciones estándar. Los ficheros pertenecientes a un paquete se identifican fácilmente comprobando el identificador de usuario. Las características y particularidades de este método son demasiado complejas para describirlas en esta sección. Puedes consultar los detalles en la receta en *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*.

6.4. Entrar al entorno chroot

Es hora de entrar en el entorno chroot para iniciar la construcción e instalar tu sistema LFS final. Como usuario `root`, ejecuta el siguiente comando para entrar a un mundo que está, en este momento, poblado sólo por las herramientas temporales.

```
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
```

La opción *`-i`* pasada al comando **env** limpiará todas las variables del chroot. Después de esto solamente se establecen de nuevo las variables `HOME`, `TERM`, `PS1` y `PATH`. La construcción *`TERM=$TERM`* establece la variable `TERM` dentro del chroot al mismo valor que tiene fuera del chroot. Dicha variable es necesaria para que funcionen correctamente programas como **vim** y **less**. Si necesitas tener presentes otras variables, como `CFLAGS` o `CXXFLAGS`, este es un buen sitio para establecerlas.

Desde este punto ya no es necesario utilizar la variable `LFS` porque todo lo que hagas estará restringido al sistema de ficheros LFS. Esto se debe a que al intérprete de comandos se le dice que `$LFS` es ahora el directorio raíz (`/`).

Advierte que `/tools/bin` queda último en el `PATH`. Esto significa que una herramienta temporal no volverá a usarse a partir de que se instale su versión final. Esto ocurre cuando el intérprete de comandos no “recuerda” la localización de los binarios ejecutados; por esta razón se desactiva la tabla interna de rutas pasándole la opción *`+h`* a **bash**.

Ten en cuenta que en la línea de entrada de comandos de **bash** pondrá: `I have no name!` (¡No tengo nombre!). Esto es normal pues el fichero `/etc/passwd` aún no ha sido creado.

Nota

Debes asegurarte de que todos los comandos que aparecen en el resto de este y los siguientes capítulos son ejecutados dentro del entorno chroot. Si por alguna razón abandonas este entorno (tras un reinicio, por ejemplo), asegurate de que los sistemas de ficheros virtuales del núcleo estan montados como se indica en Sección 6.2.2, “Montar y poblar /dev” y Sección 6.2.3, “Montar los sistemas de ficheros virtuales del núcleo” y entra de nuevo en el chroot antes de seguir con la instalación.

6.5. Creación de los directorios

Es hora de crear cierta estructura en el sistema de ficheros LFS. Crea un árbol de directorios estándar ejecutando los siguientes comandos:

```
mkdir -pv /{bin,boot,etc/opt,home,lib,mnt,opt}
mkdir -pv /{media/{floppy,cdrom},sbin,srv,var}
install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
mkdir -pv /usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv /usr/{,local/}share/{doc,info,locale,man}
mkdir -v  /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
for dir in /usr /usr/local; do
  ln -sv share/{man,doc,info} $dir
done
mkdir -v /var/{lock,log,mail,run,spool}
mkdir -pv /var/{opt,cache,lib/{misc,locate},local}
```

Los directorios se crean por defecto con los permisos 755, pero esto no es deseable para todos los directorios. En los comandos anteriores se hacen dos cambios: uno para el directorio personal de `root` y otro para los directorios de los ficheros temporales.

El primer cambio nos asegura que nadie aparte de root pueda entrar en el directorio `/root`, lo mismo que debería hacer un usuario normal con su directorio personal. El segundo cambio nos asegura que cualquier usuario pueda escribir en los directorios `/tmp` y `/var/tmp`, pero no pueda borrar los ficheros de otros usuarios. Esto último lo prohibe el llamado “bit pegajoso” (sticky bit), el bit más alto (1) en la máscara de permisos 1777.

6.5.1. Nota de conformidad con FHS

El árbol de directorios está basado en el Estándar de la Jerarquía del Sistema de Ficheros (FHS, disponible en *http://www.pathname.com/fhs/*). En adición al FHS creamos enlaces simbólicos de compatibilidad para los directorios `man`, `doc` y `info`, pues muchos paquetes todavía intentan instalar su documentación en `/usr/<directorio>` o `/usr/local/<directorio>` en vez de en `/usr/share/<directorio>` o `/usr/local/share/<directorio>`. El FHS también estipula la existencia de `/usr/local/games` y `/usr/share/games`.Como sobre la estructura del subdirectorio `/usr/local/share` el FHS no es preciso, creamos aquí sólo los directorios que son necesarios. Sin embargo, eres libre de crear esos directorios si prefieres cumplir estríctamente con el FHS.

6.6. Creación de ficheros y enlaces simbólicos esenciales

Algunos programas tienen fijadas en su código rutas a programas que aún no existen. Para satisfacer a estos programas creamos unos cuantos enlaces simbólicos que serán sustituidos por ficheros reales durante el transcurso de este capítulo a medida que vayamos instalando todos los programas:

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

Un sistema Linux correcto mantiene una lista de los sistemas de ficheros montados en `/etc/mtab`. Normalmente, este fichero se crearía al montar un nuevo sistema de ficheros. Puesto que no montaremos ningún sistema de ficheros dentro del entorno chroot, crea un fichero vacío para las utilidades que esperan que `/etc/mtab` esté presente:

```
touch /etc/mtab
```

Para que `root` pueda entrar al sistema y para que el nombre "root" sea reconocido, es necesario tener las entradas apropiadas en los ficheros `/etc/passwd` y `/etc/group`.

Crea el fichero `/etc/passwd` ejecutando el siguiente comando:

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

La contraseña real para `root` (la "x" es sólo un sustituto) se establecerá más adelante.

Crea el fichero `/etc/group` ejecutando el siguiente comando:

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

Los grupos creados no son parte de ningún estándar, son grupos escogidos en parte por los requisitos de la configuración de Udev en este capítulo, y en parte por la práctica común empleada por una serie de distribuciones Linux existentes. El LSB (Linux Standard Base, disponible en *http://www.linuxbase.org/*) sólo recomienda que, aparte del grupo `root` con GID 0, esté presente un grupo `bin` con GID 1. Todos los demás nombres de grupos y sus GID pueden ser elegidos libremente por el usuario, pues los programas correctamente escritos no dependen del número GID, sino que utilizan el nombre del grupo.

Para eliminar el “I have no name!” del símbolo del sistema, iniciaremos un nuevo intérprete de comandos. Puesto que instalamos una Glibc completa en el Capítulo 5 y acabamos de crear los ficheros `/etc/passwd` y `/etc/group`, la resolución de nombres de usuarios y grupos funcionará ahora:

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

Advierte el uso de la directiva *+h*. Esto le indica a **bash** que no utilice su tabla interna de rutas. Sin esta directiva, **bash** recordaría la ruta a los binarios que ha ejecutado. Para poder usar los binarios recién compilados tan pronto como sean instalados, se usará la directiva *+h* durante el resto de este capítulo.

Los programas **login**, **getty** e **init** (entre otros) mantienen una serie de ficheros de registro con información sobre quienes están y estaban dentro del sistema. Sin embargo, estos programas no crean dichos ficheros si no existen. Crea los ficheros de registro con sus permisos correctos:

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

El fichero `/var/run/utmp` lista los usuarios que están actualmente dentro del sistema, `/var/log/wtmp` registra todos los ingresos y salidas. El fichero `/var/log/lastlog` muestra, para cada usuario, cuando fue la última vez que ingresó, y el fichero `/var/log/btmp` lista los intentos de ingreso fallidos.

6.7. Cabeceras API de Linux-2.6.22.6

Las cabeceras API de Linux exponen la API del núcleo para ser usada por Glibc.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 286 MB

6.7.1. Instalación de las cabeceras API de Linux

El núcleo Linux necesita exponer una Interfaz de Programación de Aplicaciones (API) para ser usada por la librería C del sistema (Glibc en LFS). Esto se hace sanitizando los diversos ficheros de cabecera incluidos en el paquete con las fuentes de núcleo Linux.

Aplica una sustitución **sed** para suprimir la instalación de los ficheros de cabecera en `/usr/include/scsi`. En su lugar se utilizará la versión proporcionada por Glibc:

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

Instala los ficheros de cabecera:

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

6.7.2. Contenido de las cabeceras API de Linux

Cabeceras instaladas: /usr/include/{asm{,-generic},linux,mtd,rdma,sound}/*.h

Descripción corta

`/usr/include/{asm{,-generic},linux,mtd,rdma,sound}/*.h`	La API de las cabeceras de Linux.

6.8. Man-pages-2.64

El paquete Man-pages contiene alrededor de 3.000 páginas de manual.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 37.4 MB

6.8.1. Instalación de Man-pages

Instala Man-pages ejecutando:

```
make install
```

6.8.2. Contenido de Man-pages

Ficheros instalados: Diversas páginas de manual

Descripción corta

`páginas de manual` Describen las funciones del lenguaje de programación C, los ficheros de dispositivo importantes y los ficheros de configuración más significativos.

6.9. Glibc-2.6.1

El paquete Glibc contiene la librería C principal. Esta librería proporciona todas las rutinas básicas para la ubicación de memoria, búsqueda de directorios, abrir y cerrar ficheros, leerlos y escribirlos, manejo de cadenas, coincidencia de patrones, aritmética, etc...

Tiempo estimado de construcción: 19.5 SBU testsuite included

Espacio requerido en disco: 556 MB testsuite included

6.9.1. Instalación de Glibc

Nota

Algunos paquetes externos a LFS suguienren la instalación de GNU libiconv para poder traducir datos de una codificación a otra. La página del proyecto (*http://www.gnu.org/software/libiconv/*) dice “Esta librería proporciona una implementación `iconv()` para usarla en sistemas que no tienen una, o cuya implementación no puede convertir de/a Unicode”. Glibc proporciona una implementación `iconv()` y puede convertir de/a Unicode, por tanto libiconv no es necesaria en un sistema LFS.

El sistema de construcción de Glibc está muy bien autocontenido y se instalará perfectamente, incluso aunque nuestro fichero de especificaciones del compilador y los guiones del enlazador todavía apunten a `/tools`. No podemos ajustar las especificaciones y el enlazador antes de instalar Glibc, porque entonces las comprobaciones del autoconf de Glibc darían resultados erróneos y esto arruinaría nuestro objetivo de conseguir una construcción limpia.

El paquete glibc-libidn añade a Glibc soporte para nombres de dominio internacionalizados (IDN). Muchos programas que soportan IDN requieren la librería `libidn` completa (mira *http://www.linuxfromscratch.org/blfs/view/svn/general/libidn.html*), no este añadido. Desempaquetalo desde dentro del directorio de las fuentes de Glibc:

```
tar -xvf ../glibc-libidn-2.6.1.tar.gz
mv glibc-libidn-2.6.1 libidn
```

En la locale vi_VN.TCVN, **bash** entra en un bucle infinito al inicio. Se desconoce si esto es un fallo de **bash** o un problema de Glibc. Desactiva la instalación de dicha locale para evitar el problema:

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

Cuando se ejecuta **make install**, un guión llamado `test-installation.pl`realiza un pequeña prueba de sanidad de nuestra recién instalada Glibc. Sin embargo, debido a que nuestras herramientas principales todavía apuntan al directorio `/tools`, la prueba de sanidad podría efectuarse sobre la Glibc equivocada. Podemos forzar que el guión compruebe la Glibc recién instalada con lo siguiente:

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

El guión **ldd** contiene sintaxis específica de Bash. Canbia su programa intérprete a **/bin/bash** por si decides instalar un **/bin/sh** diferente como se describe en el capítulo *Shells* del libro BLFS:

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

La documentación de Glibc recomienda construirlo fuera del árbol de las fuentes, en un directorio de construcción dedicado:

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

Añade de nuevo a CFLAGS la opción de compilación necesaria:

```
echo "CFLAGS += -march=i486" > configparms
```

Prepara Glibc para su compilación:

```
../glibc-2.6.1/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --enable-kernel=2.6.0 --libexecdir=/usr/lib/glibc
```

Significado de la nueva opción de configure:

`--libexecdir=/usr/lib/glibc`
Esto cambia la localización del programa **pt_chown** de su ubicación por defecto `/usr/libexec` a `/usr/lib/glibc`.

Compila el paquete:

```
make
```

Importante

En esta sección, el banco de pruebas para Glibc se considera crítico. No te lo saltes bajo ninguna circunstancia.

Comprueba los resultados:

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

Posiblemente veas un fallo esperado (ignorado) en la prueba *posix/annexc*. Adicionalmente, el banco de pruebas de Glibc depende en parte del sistem anfitrión. Aquí hay una lista con los problemas más comunes:

- La prueba *nptl/tst-cancel1* falla cuando se usa la serie 4.1 de GCC.
- Las pruebas *nptl/tst-clock2* y *tst-attr3* fallan a veces. La razón no se entiende por completo, pero hay indicios de q2ue una alta carga del sistema puede probocarlos.
- Las pruebas *math* fallan en ocasiones cuando se ejecutan en sistemas donde la CPU no es una Intel o AMD genuina relativamente nueva.
- Si has montado la partición LFS con la opción *`noatime`*, la prueba *atime* fallará. Como se mencionó en Sección 2.4, “Montar la nueva partición”, no utilices la opción *`noatime`* cuando construyas un LFS.
- Cuando se ejecutan en hardware antiguo y lento, o en sistemas bajo carga, algunas pruebas pueden fallar debido a que se excede el tiempo estimado.

Aunque se trata de un mensaje inofensivo, la fase de instalación de Glibc se quejará de la ausencia de `/etc/ld.so.conf`. Evita este molesto aviso con:

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

Instala el paquete:

```
make install
```

Las locales que hacen que el sistema responda en un idioma diferente no se instalaron con el comando anterior. Ninguna locale es requerida, pero si no se encuentran algunas de ellas los bancos de pruebas de paquetes posteriores podrían saltarse pruebas importantes.

Locales individuales pueden instalarse usando el prtograma **localedef**. Por ejemplo, el primer comando **localedef** mostrado a continuación combina la definición de locale independiente del frupo de caractéres `/usr/share/i18n/locales/de_DE` con la definición de mapa de caractéres `/usr/share/i18n/charmaps/ISO-8859-1.gz` y añade el resultado al fichero `/usr/lib/locale/locale-archive`. Las siguientes instrucciones instalarán el conjunto mínimo de locales necesario para una correcta cobertura de las pruebas:

```
mkdir -pv /usr/lib/locale
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i ja_JP -f EUC-JP ja_JP
```

Adicionalmente, instala la locale para tu propio país, idioma y conjunto de caractéres.

Alternatívamente, instala todas las locales listadas en el fichero `glibc-2.6.1/localedata/SUPPORTED` (incluye todas las locales listadas arriba y muchas mas) con el siguiente comando, el cual tarda bastante tiempo en finalizar:

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

Entoces usa el comando **localedef** para crear e instalar las locales no listadas en el fichero `glibc-2.6.1/localedata/SUPPORTED` en el improbable caso de que las necesites.

6.9.2. Configuración de Glibc

Necesitamos crear el fichero `/etc/nsswitch.conf`, porque aunque Glibc nos facilita los valores por defecto cuando este fichero no se encuentra o está corrupto, estos valores por defecto no funcionan bien en un entorno de red. También hay que configurar la zona horaria.

Crea un nuevo fichero `/etc/nsswitch.conf` ejecutando lo siguiente:

```
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
```

Una forma de determinar la zona horaria local es ejecutar el siguiente guión:

```
tzselect
```

Después de contestar unas preguntas referentes a tu localización, el guión te mostrará el nombre de tu zona horaria (por ejemplo *America/Edmonton*). Otras zonas locales posibles, como *Canada/Eastern* or *EST5EDT*, que no son identificadas por el guión pero que pueden usarse, se encuentran listadas en `/usr/share/zoneinfo`.

Crea entonces el fichero `/etc/localtime` ejecutando:

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

Sustituye *<xxx>* con el nombre de la zona horaria seleccionada (por ejemplo, Europe/Madrid).

Significado de la opción de cp:

`--remove-destination`

Esto es necesario para forzar la eliminación del enlace simbólico que ya existe. La razón por la que copiamos en lugar de enlazar es para cubrir el caso en el que `/usr` está en otra partición. Esto puede ser importante cuando se arranca en modo de usuario único.

6.9.3. Configuración del cargador dinámico

Por defecto, el cargador dinámico (`/lib/ld-linux.so.2`) busca en `/lib` y `/usr/lib` las librerías dinámicas que necesitan los programas cuando los ejecutas. No obstante, si hay librerías en otros directorios que no sean `/lib` y `/usr/lib`, necesitas añadirlos al fichero de configuración `/etc/ld.so.conf` para que el cargador dinámico pueda encontrarlas. Dos directorios típicos que contienen librerías adicionales son `/usr/local/lib` y `/opt/lib`, así que añadimos estos directorios a la ruta de búsqueda del cargador dinámico.

Crea un nuevo fichero `/etc/ld.so.conf` ejecutando lo siguiente:

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

/usr/local/lib
/opt/lib

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

6.9.4. Contenido de Glibc

Programas instalados: catchsegv, gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump, and zic

Librerías instaladas: ld.so, libBrokenLocale.{a,so}, libSegFault.so, libanl.{a,so}, libbsd-compat.a, libc.{a,so}, libcidn.so, libcrypt.{a,so}, libdl.{a,so}, libg.a, libieee.a, libm.{a,so}, libmcheck.a, libmemusage.so, libnsl.{a,so}, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libnss_nis.so, libnss_nisplus.so, libpcprofile.so, libpthread.{a,so}, libresolv.{a,so}, librpcsvc.a, librt.{a,so}, libthread_db.so y libutil.{a,so}

Descripciones cortas

catchsegv Puede usarse para crear una traza de la pila cuando un programa termina con una violación de segmento.

gencat Genera catálogos de mensajes.

getconf Muestra los valores de configuración del sistema para variables específicas del sistema de ficheros.

getent Obtiene entradas de una base de datos administrativa.

iconv Realiza conversiones de juego de caracteres.

iconvconfig Crea un fichero de configuración para la carga rápida del módulo **iconv**.

ldconfig Configura las asociaciones en tiempo de ejecución para el enlazador dinámico.

ldd Muestra las librerías compartidas requeridas por cada programa o librería especificados.

lddlibc4 Asiste a **ldd** con los ficheros objeto.

locale Mustra información diversa sobre la locale actual.

localedef Compila las especificaciones de locales.

mtrace Lee e interpreta un fichero de traza de memoria y muestra un sumario en formato legible.

nscd Un demonio que suministra una caché para las peticiones más comunes al servidor de nombres.

pcprofiledump Vuelca la información generada por un perfil de PC.

pt_chown Un programa de ayuda para **grantpt** que establece el propietario, grupo y permisos de acceso para un pseudo-terminal esclavo.

rpcgen Genera código C para implementar el protocolo RPC.

rpcinfo Hace una llamada RPC a un servidor RPC.

sln Un programa **ln** enlazado estáticamente.

sprof Lee y muestra los datos del perfil de los objetos compartidos.

tzselect Pregunta al usuario información sobre la localización actual y muestra la descripción de la zona horaria correspondiente.

xtrace Traza la ejecución de un programa mostrando la función actualmente ejecutada.

zdump El visualizador de la zona horaria.

zic El compilador de la zona horaria.

`ld.so` El programa de ayuda para las librerías compartidas ejecutables.

`libBrokenLocale` Usada internamente por Glibc como un gran apaño para hacer que programas rotois (por ejemplo algunas aplicaciones Motif) funcionen. Para mas información, mira los comentarios en `glibc-2.6.1/locale/broken_cur_max.c`.

`libSegFault` El manejador de señales de violación de segmento, usado por **catchsegv**.

`libanl` Una librería de búsqueda de nombres asíncrona.

`libbsd-compat` Proporciona la portabilidad necesaria para ejecutar ciertos programas BSD en Linux.

`libc` La librería principal de C.

`libcidn` Usada internamente por Glibc para manejar nombres de dominio internacionalizados en la función `getaddrinfo()`.

`libcrypt` La librería criptográfica.

`libdl` La librería de interfaz del enlazado dinámico.

`libg` Librería vacía que no contiene funciones. Anteriormente era una librería en tiempo de ejecución para **g++**.

`libieee` Al enlazar contra este módulo se fuerzan las reglas de manejo de errores para funciones matemáticas según se define por el Instituto de Ingenieros Electricos y Electrónicos (IEEE). Por defecto se usa manejo de errores POSIX.1.

`libm` La librería matemática.

`libmcheck` Activa la comprobación de ocupación de memoria cuando se enlaza contra ella.

`libmemusage` Usada por **memusage** para ayudar a recoger información sobre el uso de memoria de un programa.

`libnsl` La librería de servicios de red.

`libnss` Las librerías Name Service Switch (Interruptor del Servicio de Nombres). Contienen funciones para resolver nombres de sistemas, usuarios, grupos, alias, servicios, protocolos y similares.

`libpcprofile` Contiene funciones de perfiles utilizadas para observar la cantidad de tiempo de CPU utilizado por líneas concretas del código fuente.

`libpthread` La librería de hilos POSIX.

`libresolv` Proporciona funciones para la creación, envío e interpretación de paquetes de datos a servidores de nombres de dominio de Internet.

`librpcsvc`	Proporciona funciones para una miscelánea de servicios RPC.
`librt`	Proporciona funciones para muchas de las interfaces especificadas por el POSIX.1b Realtime Extension (Extensiones en Tiempo Real POSIX.1b).
`libthread_db`	Contiene funciones útiles para construir depuradores para programas multihilo.
`libutil`	Contiene código para funciones “estándar” usadas en diferentes utilidades Unix.

6.10. Reajustar las herramientas

Ahora que hemos instalado las librerías de C finales, es hora de ajustar de nuevo el conjunto de herramientas. Las ajustaremos para que enlacen cualquier nuevo programa compilado contra estas nuevas librerías. Este es un proceso similar al usado en la fase "Ajustar" al principio del Capítulo 5, pero en sentido contario. En el Capítulo 5 el cambio iba de los directorios `/{,usr}/lib` del anfitrión al nuevo directorio `/tools/lib`. Ahora es guiado de `/tools/lib` a los directorios `/{,usr}/lib` del LFS.

Primero haz una copia de respaldo del enlazador situado en `/tools` y reemplazalo con el enlazador ajustado que creamos en el capítulo 5. También crearemos un enlaze a su contraparte en `/tools/$(gcc -dumpmachine)/bin`:

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

A continuación, corrige el fichero de especificaciones de GCC para que apunte al nuevo enlazador dinámico, y por tanto GCC sepa dónde encontrar las cabeceras correctas y los ficheros de inicio de Glibc. Un comando **sed** realiza esto:

Importante

Si estás trabajando sobre una plataforma en la que el nombre del enlazador dinámico no sea `ld-linux.so.2`, sustituye "ld-linux.so.2" en el comando siguiente por el nombre del enlazador dinámico para tu plataforma. Si es necesario, consulta la Sección 5.2, "Notas técnicas sobre las herramientas".

```
gcc -dumpspecs | sed \
    -e 's@/tools/lib/ld-linux.so.2@/lib/ld-linux.so.2@g' \
    -e '/\*startfile_prefix_spec:/{n;s@.*@/usr/lib/ @}' \
    -e '/\*cpp:/{n;s@$@ -isystem /usr/include@}' > \
    `dirname $(gcc --print-libgcc-file-name)`/specs
```

Es buena idea inspeccionar visualmente el fichero de especificaciones para verificar que realmente se produjeron los cambios deseados.

Importante

Si estás trabajando sobre una plataforma en la que el nombre del enlazador dinámico no sea `ld-linux.so.2`, sustituye "ld-linux.so.2" en el comando anterior por el nombre del enlazador dinámico para tu plataforma. Si es necesario, consulta la Sección 5.2, "Notas técnicas sobre las herramientas".

En este punto es obligatorio asegurarse de que las operaciones básicas (compilación y enlazado) de las nuevas herramientas ajustadas funcionan como se espera. Para hacer esto, realiza las siguientes comprobaciones de sanidad:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser (con las diferencias para la plataforma sobre el nombre del enlazador dinámico):

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

Comprueba que `/lib` aparezca como prefijo de tu enlazador dinámico.

Ahora asegurate de que lo hemos configurado para usar los ficheros de inicio correctos:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

Verifica que el compilador busca los ficheros de cabecera correctos:

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

Este comando debería mostrar la siguiente salida:

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

A continuación verifica que el nuevo enlazador se está usando con la ruta de búsqueda correcta:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

Segidamente asegurate de estar usando la libc correcta:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

Por último, asegurate de que GCC utiliza el enlazador dinámico correcto:

```
grep found dummy.log
```

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser (teniendo en cuenta las diferencias en el nombre del enlazador dinámico específico para cada plataforma):

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

Si no recibes una salida como la mostrada arriba, o no hay salida alguna, algo está realmente mal. Necesitarás investigar y revisar tus pasos para encontrar el problema y corregirlo. La razón más probable es que algo salió mal durante el anterior arreglo del fichero specs. Deberás resolver todos los problemas antes de seguir con el proceso.

Una vez que todo funcione coorectamente, borra los ficheros de prueba:

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

6.11. Binutils-2.18

El paquete Binutils contiene un enlazador, un ensamblador y otras utilidades para trabajar con ficheros objeto.

Tiempo estimado de construcción: 1.7 SBU testsuite included

Espacio requerido en disco: 186 MB testsuite included

6.11.1. Instalación de Binutils

Verifica que tus pseudo-terminales (PTYs) funcionan adecuadamente dentro del entorno chroot. Comprueba que todo está correcto realizando una simple prueba:

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

Si recibes el siguiente mensaje, el entorno chroot no está correctamente configurado para operar con PTYs:

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

El sistema no tiene más ptys.
Pídele al administrador del sistema que cree más.
```

Debes solucionar el problema antes de ejecutar los bancos de pruebas de Binutils y GCC.

La documentación de Binutils recomienda construirlo fuera del árbol de las fuentes, en un directorio de construcción dedicado:

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

Prepara Binutils para su compilación:

```
../binutils-2.18/configure --prefix=/usr \
    --enable-shared
```

Compila el paquete:

```
make tooldir=/usr
```

Significado del parámetro de make:

`tooldir=/usr`

Normalmente, tooldir (el directorio donde se instalarán los ejecutables) se establece como `$(exec_prefix)/$(target_alias)`. Por ejemplo, en máquinas i686 esto se convertiría en `/usr/i686-pc-linux-gnu`. Como este es un sistema personalizado, no es necesario tener en `/usr` dicho directorio específico de un objetivo. `$(exec_prefix)/$(target_alias)` se utilizaría si el sistema fuese usado para compilación cruzada (por ejemplo, para compilar un paquete en una máquina Intel, pero generando código que se ejecutará en máquinas PowerPC).

Importante

En esta sección, el banco de pruebas para Binutils se considera crítico. No te lo saltes bajo ninguna circunstancia.

Comprueba los resultados:

```
make check
```

Instala el paquete:

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

Instala el fichero de cabecera `libiberty`, pues lo necesitan algunos paquetes:

```
cp -v ../binutils-2.18/include/libiberty.h /usr/include
```

6.11.2. Contenido de Binutils

Programas instalados: addr2line, ar, as, c++filt, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings y strip

Librerías instaladas: libiberty.a, libbfd.{a,so} y libopcodes.{a,so}

Descripciones cortas

addr2line Traduce direcciones de programas a nombres de ficheros y números de líneas. Dándole una dirección y un ejecutable, usa la información de depuración del ejecutable para averiguar qué fichero y número de línea está asociado con dicha dirección.

ar Crea, modifica y extrae desde archivos.

as Un ensamblador que ensambla la salida de **gcc** dentro de ficheros objeto.

c++filt Es usado por el enlazador para decodificar símbolos de C++ y Java, guardando las funciones sobrecargadas para su clasificación.

gprof Muestra los datos del perfil del gráfico de llamada.

ld Un enlazador que combina un número de ficheros objeto y de archivos en un único fichero, reubicando sus datos y estableciendo las referencias a los símbolos.

nm Lista los símbolos que aparecen en un fichero objeto dado.

objcopy Traduce un tipo de ficheros objeto a otro.

objdump Muestra información sobre el fichero objeto indicado, con opciones para controlar la información a mostrar. La información mostrada es útil fundamentalmente para los programadores que trabajan sobre las herramientas de compilación.

ranlib Genera un índice de los contenidos de un archivo, y lo coloca en el archivo. El índice lista cada símbolo definido por los miembros del archivo que son ficheros objeto reubicables.

readelf Muestra información sobre binarios de tipo ELF.

size Lista los tamaños de las secciones y el tamaño total para los ficheros objeto indicados.

strings Muestra, para cada fichero indicado, las cadenas de caracteres imprimibles de al menos la longitud especificada (4 por defecto). Para los ficheros objeto muestra, por defecto, sólo las cadenas procedentes de las secciones de inicialización y carga. Para otros tipos de ficheros explora el fichero al completo.

strip Elimina símbolos de ficheros objeto.

`libiberty` Contiene rutinas usadas por varios programas GNU, incluidos **getopt**, **obstack**, **strerror**, **strtol** y **strtoul**.

`libbfd` La librería del Descriptor de Fichero Binario.

`libopcodes` Una librería para manejar mnemónicos. Se usa para construir utilidades como **objdump**. Los mnemónicos son las versiones en “texto legible” de las instrucciones del procesador.

6.12. GCC-4.2.1

El paquete GCC contiene la colección de compiladores GNU, que incluye los compiladores C y C++.

Tiempo estimado de construcción:	22 SBU testsuite included
Espacio requerido en disco:	681 MB testsuite included

6.12.1. Instalación de GCC

Aplica una sustitución **sed** que suprimirá la instalación de `libiberty.a`. Se usará en su lugar la versión de `libiberty.a` suministrada por Binutils:

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

El proceso "bootstrap" realizado en Sección 5.4, “GCC-4.2.1 - Fase 1” construye GCC con la opción `-fomit-frame-pointer`. Las construcciones no "bootstrap" omiten dicha opción, así que aplica el siguiente **sed** para usarla y asegurar construcciones consistentes del compliador:

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

Se sabe que el guión **fixincludes** en ocasiones intenta “corregir” erroneamente las cabeceras instaladas anteriormente en el sistema. Como se sabe que las cabeceras instaladas por GCC-4.2.1 y Glibc-2.6.1 no necesitan corrección, ejecuta el siguiente comando para evitar que se ejecute el guión **fixincludes**:

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

GCC proporciona un guión **gccbug** que detecta en tiempo de compilación si **mktemp** está presente, y fija el resultado en una prueba. Si no lo encuentra, el guión utilizará nombres menos aleatorios para los ficheros temporales. Instalaremos Mktemp mas tarde, por lo que el siguiente **sed** simulará su presencia:

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

La documentación de GCC recomienda construirlo fuera del árbol de las fuentes, en un directorio de construcción dedicado:

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

Prepara GCC para su compilación:

```
../gcc-4.2.1/configure --prefix=/usr \
    --libexecdir=/usr/lib --enable-shared \
    --enable-threads=posix --enable-__cxa_atexit \
    --enable-clocale=gnu --enable-languages=c,c++ \
    --disable-bootstrap
```

Compila el paquete:

```
make
```

Importante

En esta sección, el banco de pruebas para GCC se considera crítico. No te lo saltes bajo ninguna circunstancia.

Comprueba los resultados, pero no pares en los errores:

```
make -k check
```

Para ver un resúmen del resultado de las pruebas, ejecuta:

```
../gcc-4.2.1/contrib/test_summary
```

Pare ver sólo los resultados, tuneliza la salída a través de **`grep -A7 Summ`**.

Los resultados pueden compararse con los que se encuentran en *http://www.linuxfromscratch.org/lfs/build-logs/development/*.

Algunos fallos inexperados no pueden evitarse siempre. Los desarrolladores de GCC están normalmente enterados de estos problemas, pero no los han resuleto aún. En concreto, se sabe que las pruebas de `libmudflap` son particularmente problemáticas debido a un error en GCC (*http://gcc.gnu.org/bugzilla/show_bug.cgi?id=20003*). A no ser que los resultados de las pruebas varíen notablemente de los mostrados en la URL anterior, puedes continuar tranquilo.

Instala el paquete:

```
make install
```

Algunos paquetes esperan que el preprocesador de C esté instalado en el directorio `/lib`. Para dar soporte a estos paquetes, crea un enlace simbólico:

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

Muchos paquetes usan el nombre **cc** para llamar al compilador C. Para satisfacer a dichos paquetes, crea un enlace simbólico:

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

Ahora que nuestras herramientas principales finales están en su sitio, es importante asegurarse de nuevo de que la compilación y el enlazado funcionan como se espera. Para hacer esto realizaremos las mismas comprobaciones de sanidad que usamos anteriormente en este capítulo:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser (con las diferencias para la plataforma sobre el nombre del enlazador dinámico):

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

Ahora asegurate de que lo hemos configurado para usar los ficheros de inicio correctos:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

```
/usr/lib/gcc/i686-pc-linux-gnu/4.2.1/../../../crt1.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.2.1/../../../crti.o succeeded
/usr/lib/gcc/i686-pc-linux-gnu/4.2.1/../../../crtn.o succeeded
```

Verifica que el compilador busca los ficheros de cabecera correctos:

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

Este comando debería mostrar la siguiente salida:

```
#include <...> search starts here:
 /usr/local/include
 /usr/lib/gcc/i686-pc-linux-gnu/4.2.1/include
 /usr/include
```

A continuación verifica que el nuevo enlazador se está usando con la ruta de búsqueda correcta:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

Segidamente asegurate de estar usando la libc correcta:

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

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser:

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

Por último, asegurate de que GCC utiliza el enlazador dinámico correcto:

```
grep found dummy.log
```

Si todo funciona correctamente, no debe haber errores y la salida del último comando debe ser (teniendo en cuenta las diferencias en el nombre del enlazador dinámico específico para cada plataforma):

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

Si no recibes una salida como la mostrada arriba, o no hay salida alguna, algo está realmente mal. Necesitarás investigar y revisar tus pasos para encontrar el problema y corregirlo. La razón más probable es que algo salió mal durante el anterior arreglo del fichero specs. Deberás resolver todos los problemas antes de seguir con el proceso.

Una vez que todo funcione coorectamente, borra los ficheros de prueba:

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

6.12.2. Contenido de GCC

Programas instalados: c++, cc (enlace a gcc), cpp, g++, gcc, gccbug y gcov
Librerías instaladas: libgcc.a, libgcc_eh.a, libgcc_s.so, libmudflap.{a,so}, libssp.{a,so}, libstdc++.{a,so} y libsupc++.a

Descripciones cortas

c++ El compilador de C++.

cc El compilador de C.

cpp El preprocesador de C. Lo usa el compilador para expandir las sentencias #include, #define y similares en los ficheros fuente.

g++ El compilador de C++.

gcc El compilador de C.

gccbug Un guión del intérprete de comandos que ayuda a crear notificaciones de errores.

gcov Una herramienta para pruebas de rendimiento. Se usa para analizar programas y encontrar qué optimizaciones tendrán el mayor efecto.

`libgcc` Contienen el soporte en tiempo de ejecución para **gcc**.

`libmudflap` Contiene rutinas para la comprobación de límites de funcionalidad de GCC.

`libssp` Contiene rutinas de soporte para la funcionalidad de protección stack-smashing de GCC.

`libstdc++` La librería estándar de C++.

`libsupc++` Proporciona rutinas de soporte para el lenguaje de programación c++.

6.13. Berkeley DB-4.6.19

El paquete Berkeley DB contiene programas y utilidades usadas por muchas otras aplicaciones para funciones relacionadas con bases de datos.

Tiempo estimado de construcción: 1.2 SBU
Espacio requerido en disco: 77 MB

Otras posibilidades de instalación

En el libro BLFS hay instrucciones para construir este paquete si necesitas construir el servidor RPC o enlaces de lenguaje adicionales. Los enlaces de lenguaje adicionales requieren paquetes adicionales. Las instrucciones de instalación sugeridas por BLFS se encuentran en *http://www.linuxfromscratch.org/blfs/view/svn/server/databases.html#db*.

Igualmente, GDBM *podría* usarse en vez de Berkeley DB para satisfacer a Man-DB. Sin embargo, como Berkeley DB está considerado como parte integrante de la construcción del LFS, este no será listado como dependencia para ningún paquete del libro BLFS. Del mismo modo, se han dedicado muchas horas para probar LFS con Berkeley DB instalado, no con GDBM. Si comprendes por completo los riesgos y los beneficios de usar GDBM y de todas formas deseas usarlo, consulta las instrucciones del BLFS que se encuentran en *http://www.linuxfromscratch.org/blfs/view/svn/general/gdbm.html*

6.13.1. Instalación de Berkeley DB

Prepara Berkeley DB para su compilación:

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

Significado de las opciones de configure:

`--enable-compat185`
Esta opción activa la construcción de la API de compatibilidad con Berkeley DB 1.85.

`--enable-cxx`
Esta opción activa la construcción de las librerías para la API de C++.

Compila el paquete:

```
make
```

No es posible testear correctamente el paquete, pues esto depende de la construcción de los enlaces TCL. Los enlaces TCL no pueden construirse ahora debido a que TCL está enlazado contra la Glibc que hay en `/tools`, no contra la Glibc de `/usr`.

Instala el paquete:

```
make docdir=/usr/share/doc/db-4.6.19 install
```

Significado del parámetro de make:

`docdir=...`
Este variable especifica el lugar correcto para la documentación.

Corrige la propiedad de la documentación instalada:

```
chown -Rv root:root /usr/share/doc/db-4.6.19
```

6.13.2. Contenido de Berkeley DB

Programas instalados: db_archive, db_checkpoint, db_deadlock, db_dump, db_hotbackup, db_load, db_printlog, db_recover, db_stat, db_upgrade y db_verify
Librerías instaladas: libdb.{so,a} y libdb_cxx.{so,a}

Descripciones cortas

db_archive	Imprime la ruta de los ficheros de registro que no están en uso.
db_checkpoint	Un demonio usado para monitorizar y comprobar registros de bases de datos.
db_deadlock	Se usa para abortar peticiones de bloqueo cuando se detectan interbloqueos.
db_dump	Convierte ficheros de bases de datos a un formato de fichero "plano" legible por **db_load**.
db_hotbackup	Crea capturas de “copias de respaldo en caliente” o “recuperación de fallos en caliente” de las bases de datos Berkeley DB.
db_load	Se usa para crear ficheros de bases de datos a partir de ficheros en texto plano
db_printlog	Convierte ficheros de registro de bases de datos a texto legible por humanos.
db_recover	Se usa para restaurar una base de datos a un estado consistente despues de un fallo.
db_stat	Mustra las estadísticas de las bases de datos Berkeley.
db_upgrade	Se usa para actualizar los ficheros de bases de datos a una nueva versión de Berkeley DB.
db_verify	Se usa para realizar comprobaciones de consistencia en ficheros de bases de datos.
`libdb.{so,a}`	Contiene funciones para manipular bases de datos desde programas C.
`libdb_cxx.{so,a}`	Contiene funciones para manipular bases de datos desde programas C++.

6.14. Sed-4.1.5

El paquete Sed contiene un editor de flujos.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 6.4 MB

6.14.1. Instalación de Sed

Prepara Sed para su compilación:

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

Significado de la nueva opción de comfigure:

`--enable-html`
Esto construye la documantación HTML.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.14.2. Contenido de Sed

Programa instalado: sed

Descripción corta

sed Se usa para filtrar y transformar ficheros de texto en una sola pasada.

6.15. E2fsprogs-1.40.2

El paquete E2fsprogs contiene las utilidades para manejar el sistema de ficheros `ext2`. También soporta los sistemas de ficheros `ext3` con registro de transacciones.

Tiempo estimado de construcción: 0.4 SBU
Espacio requerido en disco: 31.2 MB

6.15.1. Instalación de E2fsprogs

Corrige una ruta interna a `/bin/rm` en el banco de pruebas de E2fsprogs:

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

La documentación de E2fsprogs recomienda construir el paquete en un subdirectorio del árbol de las fuentes:

```
mkdir -v build
cd build
```

Prepara E2fsprogs para su compilación:

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

Significado de las opciones de configure:

`--with-root-prefix=""`
Ciertos programas (como el programa **e2fsck**) se consideran esenciales. Cuando, por ejemplo, `/usr` no está montado, estos programas esenciales deben estar disponibles. Pertenecen a directorios como `/lib` y `/sbin`. Si no se le pasase esta opción al configure de E2fsprogs, los programas se instalarían en el directorio `/usr`.

`--enable-elf-shlibs`
Esto crea las librerías compartidas utilizadas por algunos de los programas de este paquete.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Una de las pruebas de E2fsprogs intenta ubicar 256 MB Si no tienes una memoria RAM mayor que esta, es recomendable que actives el espacio suficiente de memoria de intercambio para la prueba. Mira en Sección 2.3, “Crear un sistema de ficheros en la partición” y Sección 2.4, “Montar la nueva partición” los detalles para crear y activar un espacio de intercambio.

Instala los binarios, la documentación y las librerías compartidas:

```
make install
```

Instala las librerías estáticas y las cabeceras:

```
make install-libs
```

6.15.2. Contenido de E2fsprogs

Programas instalados: badblocks, blkid, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, filefrag, findfs, fsck, fsck.ext2, fsck.ext3, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs y uuidgen.

Librerías instaladas: libblkid.{a,so}, libcom_err.{a,so}, libe2p.{a,so}, libext2fs.{a,so}, libss.{a,so} y libuuid.{a,so}

Descripciones cortas

badblocks Busca bloques dañados en un dispositivo (normalmente una partición de disco).

blkid Una utilidad de línea de comandos para localizar y mostrar atributos de dispositivos de bloque.

chattr Cambia los atributos de los ficheros en un sistema de ficheros `ext2` y también en sistemas de ficheros `ext3`, la versión con registro de transacciones del sistema de ficheros `ext2`.

compile_et Un compilador de tablas de error. Convierte una tabla de códigos de error y mensajes en un fichero fuente C apropiado para usar con la librería `com_err`.

debugfs Un depurador de sistemas de ficheros. Puede usarse para examinar y cambiar el estado de un sistema de ficheros `ext2`.

dumpe2fs Muestra la información del superbloque y de los grupos de bloques del sistema de ficheros presente en un determinado dispositivo.

e2fsck Se usa para chequear, y opcionalmente reparar, sistemas de ficheros `ext2` y también `ext3`.

e2image Se usa para salvar información crítica de un sistema de ficheros `ext2` en un fichero.

e2label Muestra o cambia la etiqueta de un sistema de ficheros `ext2` situado en el dispositivo especificado.

filefrag Informa sobre lo mal fragmentado que puede estar un sistema de ficheros en concreto.

findfs Encuentra un sistema de ficheros por su etiqueta o UUID (Identificador Universal Único).

fsck Se usa para chequear, y opcionalmente reparar, un sistema de ficheros. Por defecto comprueba los sistemas de ficheros listados en `/etc/fstab`.

fsck.ext2 Por defecto comprueba sistema de ficheros `ext2`.

fsck.ext3 Por defecto comprueba sistemas de ficheros `ext3`.

logsave Salva la salida de un comando en un fichero de registro.

lsattr Muestra los atributos de un fichero en un sistema de ficheros `ext2`.

mk_cmds Convierte una tabla de nombres de comandos y mensajes de ayuda en un fichero fuente C preparado para usarlo con la librería del subsistema `libss`.

mke2fs Crea un sistema de ficheros `ext2` o `ext3` en un dispositivo dado.

mkfs.ext2 Por defecto crea un sistema de ficheros `ext2`.

mkfs.ext3 Por defecto crea un sistema de ficheros `ext3`.

mklost+found Se usa para crear un directorio `lost+found` en un sistema de ficheros `ext2`. Reserva bloques de disco para este directorio facilitando la tarea de **e2fsck**.

resize2fs Se usa para redimensionar sistemas de ficheros `ext2`.

tune2fs Ajusta los parámetros de un sistema de ficheros `ext2`.

uuidgen	Crea un nuevo UUID. Cada nuevo UUID puede considerarse razonablemente único por muchos UUID que se hayan creado en el sistema local o en otros sistemas en el pasado o en el futuro.
`libblkid`	Contiene rutinas para la identificación de dispositivos y extracción de marcas.
`libcom_err`	Rutina para mostrar errores comunes.
`libe2p`	Usada por **dumpe2fs**, **chattr** y **lsattr**.
`libext2fs`	Contiene rutinas para permitir a los programas de nivel de usuario manipular un sistema de ficheros `ext2`.
`libss`	Usada por **debugfs**.
`libuuid`	Contiene rutinas para generar identificadores únicos para objetos que pueden estar accesibles más allá del sistema local.

6.16. Coreutils-6.9

El paquete Coreutils contiene utilidades para mostrar y establecer las características básicas del sistema.

Tiempo estimado de construcción: 1.0 SBU
Espacio requerido en disco: 72.4 MB

6.16.1. Instalación de Coreutils

La versión de la función “futimens” usada por Coreutils es incompatible con la versión actual proporcionada por Glibc. Por tanto, renombra la función:

```
sed -i 's/futimens/gl_&/' src/{copy,touch}.c lib/utimens.{c,h}
```

Un problema conocido en el programa **uname** de este paquete es que la opción *-p* siempre devuelve `unknown` (desconocido). El siguiente parche corrige este comportamiento en arquitecturas Intel:

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

Evita que Coreutils instale binarios que serán instalados más tarde por otros paquetes:

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

POSIX requiere que los programas de Coreutils reconozcan correctamente la amplitud de los caractéres incluso en locales multibytes. El siguiente parche corrigue este incumplimiento y otros errores relacionados con la internacionalización:

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

Para poder superar la prueba añadida por este parche, deben cambierse los permisos del fichero de la prueba:

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

Nota

En el pasado se encontraron muchos errores en dicho parche. Cuando informes de nuevos fallos a los mantenedores de Coreutils, comprueba primero si estos son reproducibles sin el parche.

Prepara Coreutils para su compilación:

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

Compila el paquete:

```
make
```

Si decides no ejecutar el banco de pruebas, salta hasta “Instala el paquete”.

Ahora todo está preparado para ejecutar el banco de pruebas. Primero ejecuta las pruebas que requieren que se ejecuten como `root`:

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

Vamos a ejecutar el resto de pruebas como usuario nobody. Sin embargo, algunas pruebas necesitan que el usuario sea miembro de más de un grupo. Para que estas pruebas no sean saltadas añadiremos un grupo temporal y haremos al usuario nobody miembro suyo:

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

Ejecuta las pruebas:

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

Elimina el grupo temporal:

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

Instala el paquete:

```
make install
```

Mueve los programas a la localización especificada por el FHS:

```
mv -v /usr/bin/{cat,chgrp,chmod,chown,cp,date,dd,df,echo} /bin
mv -v /usr/bin/{false,hostname,ln,ls,mkdir,mknod,mv,pwd,readlink,rm} /bin
mv -v /usr/bin/{rmdir,stty,sync,true,uname} /bin
mv -v /usr/bin/chroot /usr/sbin
```

Algunos de los guiones del paquete LFS-Bootscripts dependen de **head**, **sleep** y **nice**. Como /usr puede no estar disponible en las primeras fases del arranque, es necesario que estos binarios se encuentren en la partición raíz:

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

6.16.2. Contenido de Coreutils

Programas instalados: base64, basename, cat, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, hostname, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mv, nice, nl, nohup, od, paste, pathchk, pinky, pr, printenv, printf, ptx, pwd, readlink, rm, rmdir, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stty, sum, sync, tac, tail, tee, test, touch, tr, true, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami y yes

Descripciones cortas

base64 Codifica y decodifica datos según la especificación base64 (RFC 3548).

basename Elimina cualquier ruta y sufijo indicado de un nombre de fichero.

cat Concatena ficheros en la salida estándar.

chgrp Cambia el grupo propietario de ficheros y directorios.

chmod Cambia los permisos de cada fichero dado al modo indicado. El modo puede ser una representación simbólica de los cambios a hacer o un número octal que representa los nuevos permisos.

chown Cambia el usuario y/o el grupo propietario de ficheros y directorios.

chroot Ejecuta un comando usando el directorio especificado como directorio /.

cksum Muestra la suma de comprobación CRC (Comprobación Cíclica Redundante) y cuenta los bytes de cada fichero especificado.

comm Compara dos ficheros ordenados, sacando en tres columnas las líneas que son únicas y las líneas que son comunes.

cp Copia ficheros.

csplit Trocea un fichero en varios nuevos ficheros, separándolos de acuerdo a un patrón indicado o a un número de líneas, y muestra el número de bytes de cada nuevo fichero.

cut Imprime fragmentos de líneas, seleccionando los fragmentos de acuerdo a los campos o posiciones indicadas.

date Muestra la fecha y hora actual en un formato determinado o establece la fecha y hora del sistema.

dd Copia un fichero usando el tamaño y número de bloques indicado, mientras que, opcionalmente, realiza conversiones en él.

df Muestra la cantidad de espacio disponible (y usado) en todos los sistemas de ficheros montados, o solo del sistema de ficheros en el que se encuentran los ficheros seleccionados.

dir Lista el contenido del directirio indicado (lo mismo que **ls**).

dircolors Imprime comandos para modificar la variable de entorno `LS_COLOR`, para cambiar el esquema de color usado por **ls**.

dirname Elimina los sufijos que no son directorios del nombre de un fichero.

du Muestra la cantidad de espacio en disco usado por el directorio actual o por cada directorio indicado (incluyendo todos sus subdirectorios) o por cada fichero indicado.

echo Muestra la cadena indicada.

env Ejecuta un programa en un entorno modificado.

expand Convierte las tabulaciones a espacios.

expr Evalúa expresiones.

factor Muestra los factores primos de los números enteros especificados.

false No hace nada, infructuoso. Siempre termina con un código de estado que indica un fallo.

fmt Reformatea cada párrafo de los ficheros especificados.

fold Reajusta la longitud de línea en cada fichero dado.

groups Muestra los grupos a los que pertenece un usuario.

head Imprime las 10 primeras líneas (o el número de líneas indicado) de un fichero.

hostid Muestra el identificador numérico (en hexadecimal) de la máquina actual.

hostname Muestra o establece el nombre de la máquina actual.

id Muestra los identificadores efectivos de usuario y grupo, y los grupos a los que pertenece, del usuario actual o de un usuario dado.

install Copia ficheros mientras establece sus permisos y, si es posible, su propietario y grupo.

join Une a partir de dos ficheros las líneas que tienen campos de unión idénticos.

link Crea un enlace duro con el nombre indicado de un fichero dado.

ln Crea enlaces duros o blandos (simbólicos) entre ficheros.

logname Muestra el nombre de acceso del usuario actual.

ls Lista el contenido de cada directorio indicado.

md5sum Muestra o verifica sumas de comprobación MD5 (Mensaje de Resúmen 5).

mkdir Crea directorios con los nombres indicados.

mkfifo Crea tuberías (FIFO, el primero en entrar, el primero en salir) con los nombres indicados.

mknod Crea nodos de dispositivos con los nombres indicados. Un nodo de dispositivo es un fichero especial de caracteres o un fichero especial de bloques o una tubería.

mv Mueve o renombra ficheros o directorios.

nice Ejecuta un programa con una prioridad distinta.

nl Numera las líneas de los ficheros dados.

nohup Ejecuta un comando que no se interrumpe cuando se cierra la sesión, con su salida redirigida a un fichero de registro.

od Vuelca ficheros en octal y otros formatos.

paste Mezcla los ficheros indicados, uniendo secuencialmente las líneas correspondientes de uno y otro, separándolas con tabulaciones.

pathchk Comprueba si los nombres de ficheros son válidos o portables.

pinky Es un cliente finger ligero. Muestra algo de información sobre un determinado usuario.

pr Pagina y encolumna el texto de un fichero para imprimirlo.

printenv Muestra el entorno.

printf Muestra los argumentos dados de acuerdo al formato indicado. Muy parecido a la función printf de C.

ptx Genera un índice permutado de los contenidos de un fichero, con cada palabra clave en su contexto.

pwd Muestra el nombre del directorio de trabajo actual.

readlink Muestra el valor del enlace simbólico indicado.

rm Elimina ficheros o directorios.

rmdir Elimina directorios si están vacíos.

seq Muestra una secuencia de números, dentro de un cierto rango y con un cierto incremento.

sha1sum Muestra o verifica sumas de comprobación SHA1 de 160 bits.

sha224sum Muestra o verifica sumas de comprobación SHA de 224 bits.

sha256sum Muestra o verifica sumas de comprobación SHA de 256 bits.

sha384sum Muestra o verifica sumas de comprobación SHA de 384 bits.

sha512sum Muestra o verifica sumas de comprobación SHA de 512 bits.

shred Sobreescribe los ficheros indicados repetidamente con patrones extraños, haciendo realmente difícil recuperar los datos.

shuf Mezcla líneas de texto.

sleep Hace una pausa por el tiempo indicado.

sort Ordena las líneas de los ficheros indicados.

split Divide un fichero en trozos, por tamaño o por número de líneas.

stat Muestra el estado de ficheros o sistemas de ficheros.

stty Establece o muestra los ajuste de línea del terminal.

sum Muestra la suma de comprobación y el número de bloques para cada fichero dado.

sync Refresca los almacenadores intermedios de los sistemas de ficheros. Fuerza el guardado de los bloques modificados al disco y actualiza el superbloque.

tac Concatena en orden inverso los ficheros indicados.

tail Imprime las últimas 10 líneas (o el número de líneas indicado) de cada fichero dado.

tee Lee de la entrada estándar y escribe tanto en la salida estándar como en los ficheros indicados.

test Comprueba el tipo de los ficheros y compara valores.

touch Cambia las fechas de modificación o acceso de cada fichero especificado, poniéndole la fecha actual. Si un fichero no existe crea uno vacío.

tr Convierte, altera y borra caracteres de la entrada estándar.

true No hace nada, conseguido. Siempre termina con un código de estado que indica éxito.

tsort Realiza una ordenación topológica. Escribe una lista totalmente ordenada de acuerdo con el orden parcial del fichero especificado.

tty Muestra el nombre de fichero del terminal conectado a la entrada estándar.

uname Muestra información del sistema.

unexpand Convierte los espacios en tabulaciones.

uniq Elimina líneas consecutivas duplicadas.

unlink Elimina el fichero indicado.

users Muestra los nombres de los usuarios conectados actualmente.

vdir Es lo mismo que **ls -l**.

wc Muestra el número de líneas, palabras y bytes de un fichero, y una línea con el total si se ha especificado más de uno.

who Muestra quién está conectado.

whoami Muestra el nombre de usuario asociado con el identificador de usuario efectivo actual.

yes Muestra en pantalla “y” o una cadena de texto dada indefinidamente, hasta que es matado.

6.17. Iana-Etc-2.20

El paquete Iana-Etc contiene datos de servicios y protocolos de red.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 2.1 MB

6.17.1. Instalación de Iana-Etc

El siguiente comando convierte los datos crudos proporcionados por IANA a formatos correctos para los ficheros de datos `/etc/protocols` y `/etc/services`:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.17.2. Contenido de Iana-Etc

Ficheros instalados: /etc/protocols y /etc/services

Descripciones cortas

`/etc/protocols` Describe los diversos protocolos DARPA para Internet que están disponibles para el subsistema TCP/IP.

`/etc/services` Proporciona un mapeado entre los nombres familiares de los servicios de Internet y los números de puerto y tipo de protocolo que tienen asignados.

6.18. M4-1.4.10

El paquete M4 contiene un procesador de macros.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 5 MB

6.18.1. Instalación de M4

Prepara M4 para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.18.2. Contenido de M4

Programa instalado: m4

Descripción corta

m4 Copia los ficheros dados expandiendo en el proceso las macros que contengan. Estas macros pueden ser internas o definidas por el usuario y pueden tomar cualquier número de argumentos. Además de hacer la expansión de macros, **m4** tiene funciones internas para incluir los ficheros indicados, lanzar comandos UNIX, hacer aritmética entera, manipular texto de diversas formas, recursión, etc. El programa **m4** puede ser usado como interfaz para un compilador o como procesador de macros por sí mismo.

6.19. Bison-2.3

El paquete Bison contiene un generador de analizadores sintácticos.

Tiempo estimado de construcción:	0.2 SBU
Espacio requerido en disco:	12.3 MB

6.19.1. Instalación de Bison

Prepara Bison para su compilación:

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

El sistema de configuración hace que Bison se construya sin soporte de internacionalización en los mensajes error si un programa **bison** no se encuentra ya en el $PATH. La siguiente adición corregirá esto:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.19.2. Contenido de Bison

Programas instalados:	bison y yacc
Librería instalada:	liby.a

Descripciones cortas

bison	Genera, a partir de una serie de reglas, un programa para analizar la estructura de ficheros de texto. Bison es un sustituto de Yacc (Yet Another Compiler Compiler, Otro Compilador de Compiladores).
yacc	Un envoltorio para **bison**, destinado a los programas que todavía llaman a **yacc** en lugar de a **bison**. Invoca a **bison** con la opción *`-y`*.
`liby.a`	La librería Yacc que contiene la implementación de las funciones `yyerror` y `main` compatibles con Yacc. Esta librería normalmente no es muy útil, pero POSIX la solicita.

6.20. Ncurses-5.6

El paquete Ncurses contiene librerías para el manejo de pantallas de caracteres independiente del terminal.

Tiempo estimado de construcción: 0.7 SBU
Espacio requerido en disco: 31 MB

6.20.1. Instalación de Ncurses

Aplica el siguiente parche para corregis una serie de problemas descubiertos por la herramienta Coverity de análisis estático de código:

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

Prepara Ncurses para su compilación:

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

Significado de las opciones de configure:

`--enable-widec`
Esta opción hace que se construyan las librerías de ancho del caracter (es decir, `libncursesw.so.5.6`) en vez de las normales (o sea, `libncurses.so.5.6`). Estas librerías de ancho del caracter son usables tanto en locales multibyte como en las tradicionales de 8-bit, mientras que las librerías normales sólo funcionan correctamente en locales de 8-bit. Las librerías de ancho de caracter y las normales sn compatibles a nivel de de las fuentes, pero no a nivel binario.

Compila el paquete:

```
make
```

Este paquete tiene un banco de pruebas, pero este sólo puede ejecutarse despues de instalar el paquete. El banco se encuentra en el directorio `test/`. Para mas detalles, mira el fichero `README` de dicho directorio.

Instala el paquete:

```
make install
```

Corrige los permisos de una librería que no debería ser ejecutable:

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

Mueve las librerías al directorio `/lib`, que es donde se espera que residan:

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

Debido a que se han movido las librerías, un enlace simbólico apunta a un fichero que no existe. Regenera ese enlace simbólico:

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

Muchas aplicaciones todavía esperan que el enlazador sea capaz de encontrar las librerías Ncurses que no son de ancho del caracter. Engaña a dichas aplicaciones para que se enlacen contra las librerías de ancho del caracter mediante enlaces simbólicos y guiones de enlazado:

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

Por último, asegurate de que las aplicaciones antiguas que buscan `-lcurses` durante su construcción son aún compilables:

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

Nota

Las instrucciones anteriores no crean las librerías Ncurses que no son de ancho del caracter, pues ningún paquete compilado desde las fuentes podria enlazarse contra ellas en tiempo de ejecución. Si necesitas tener dichas librerías debido a que las necesita alguna aplicación de la que sólo tengas los ejecutables, construyelas con los siguientes comandos:

```
make distclean
./configure --prefix=/usr --with-shared --without-normal \
  --without-debug --without-cxx-binding
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

6.20.2. Contenido de Ncurses

Programas instalados: captoinfo (enlace a tic), clear, infocmp, infotocap (enlace a tic), ncurses5-config, reset (enlace a tset), tack, tic, toe, tput y tset

Librerías instaladas: libcursesw.{a,so} (enlace y guión de enlazado libncursesw.{a,so}), libformw.{a,so}, libmenuw.{a,so}, libncurses++w.a, libncursesw.{a,so}, libpanelw.{a,so} y sus correspondientes sin la "w" en el nombre de la librería para las que no son de ancho del caracter.

Descripciones cortas

captoinfo Convierte una descripción termcap en una descripción terminfo.

clear Limpia la pantalla si es posible.

infocmp Compara o imprime en pantalla una descripción terminfo.

infotocap Convierte una descripción terminfo en una descripción termcap.

ncurses5-config	Proporciona información sobre la configuración de ncurses
reset	Reinicializa un terminal a sus valores por defecto.
tack	El comprobador de acciones terminfo. Se usa principalmente para verificar la precisión de una entrada de la base de datos terminfo.
tic	El compilador de entradas de descripciones terminfo. Transforma un fichero terminfo en formato fuente al formato binario requerido por las rutinas de las librerías ncurses. Los ficheros terminfo contienen información sobre las capacidades de un terminal.
toe	Lista todos los tipos de terminal disponibles, dando el nombre primario y la descripción de cada uno.
tput	Pone a disposición del intérprete de comandos la información sobre las capacidades dependientes del terminal. También sirve para inicializar o restablecer el terminal, o para devolver su nombre largo.
tset	Sirve para inicializar terminales.
`libcurses`	Enlace a `libncurses`
`libncurses`	Contienen funciones para mostrar texto de formas complicadas en la pantalla de un terminal. Un buen ejemplo del uso de estas funciones es el menú que se muestra en el proceso **make menuconfig** del núcleo.
`libform`	Contienen funciones para implementar formularios.
`libmenu`	Contienen funciones para implementar menús.
`libpanel`	Contienen funciones para implementar paneles.

6.21. Procps-3.2.7

El paquete Procps contiene programas para monitorizar procesos.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 2.3 MB

6.21.1. Instalación de Procps

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.21.2. Contenido de Procps

Programas instalados: free, kill, pgrep, pkill, pmap, ps, pwdx, skill, slabtop, snice, sysctl, tload, top, uptime, vmstat, w y watch
Librería instalada: libproc.so

Descripciones cortas

free Muestra la cantidad total de memoria libre y usada en el sistema, tanto física como de intercambio (swap).
kill Envía señales a los procesos.
pgrep Visualiza procesos basándose en su nombre u otros atributos
pkill Envía señales a procesos basándose en su nombre u otros atributos
pmap Muestra el mapa de memoria del proceso indicado.
ps Facilita una instantánea de los procesos actuales.
pwdx Informa sobre el directorio de trabajo actual de un proceso.
skill Envía señales a procesos que coincidan con un criterio dado.
slabtop Mustra información detallada en tiempo real de la zona de intercambio del núcleo.
snice Cambia la prioridad de planificación de los procesos que coincidan con un criterio dado.
sysctl Modifica los parámetros del núcleo en tiempo de ejecución.
tload Imprime un gráfico de la carga promedio actual del sistema.
top Muestra los procesos más activos en uso de CPU. Proporciona una vista dinámica de la actividad de los procesos en tiempo real.
uptime Muestra cuánto tiempo hace que el sistema está en ejecución, cuántos usuarios están conectados y la carga media del sistema.
vmstat Muestra estadísticas de la memoria virtual, dando información sobre los procesos, memoria, paginación, entrada/salida por bloques y actividad del procesador.

w Muestra qué usuarios hay actualmente en el sistema, en qué terminal y desde cuándo.

watch Ejecuta un comando repetidamente, mostrando su primera salida a pantalla completa. Esto te permite observar los cambios en la salida al pasar el tiempo.

`libproc` Contiene funciones usadas por la mayoría de los programas de este paquete.

6.22. Libtool-1.5.24

El paquete Libtool contiene el guión de GNU para soporte genérico de librerías. Oculta la complejidad del uso de librerías compartidas tras una interfaz consistente y portable.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 16.6 MB

6.22.1. Instalación de Libtool

Prepara Libtool para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.22.2. Contenido de Libtool

Programas instalados: libtool y libtoolize
Librerías instaladas: libltdl.{a,so}

Descripciones cortas

libtool Proporciona servicios de soporte generalizados para la compilación de librerías.

libtoolize Proporciona una forma estándar de añadir soporte para **libtool** a un paquete.

`libltdl` Oculta las diversas dificultades para abrir la carga dinámica de las librerías.

6.23. Perl-5.8.8

El paquete Perl contiene el Lenguaje Práctico de Extracción e Informe.

Tiempo estimado de construcción: 1.5 SBU
Espacio requerido en disco: 143 MB

6.23.1. Instalación de Perl

Crea primero un fichero `/etc/hosts` básico que es referenciado por uno de los ficheros de configuración de Perl y también por el banco de pruebas opcional:

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

Corrige una incompatibilidad con gcc-4.2.1:

```
sed -i 's/command /command[ -]/' makedepend.SH
```

Para tener un control total sobre la forma en que Perl se configura, puedes ejecutar el guión interactivo **Configure** y modificar a mano el modo en el que se construye este paquete. Si lo prefieres, puedes usar los valores autodetectados preparando Perl para su compilación con:

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

Significado de la opción de configure:

`-Dpager="/usr/bin/less -isR"`
Esto corrige un error en el modo en que **perldoc** invoca al programa **less**.

`-Dman1dir=/usr/share/man/man1 -Dman3dir=/usr/share/man/man3`
Puesto que aún no se ha instalado Groff, **Configure** piensa que no queremos las páginas de manual de Perl. La ejecución de estos parámetros evita dicha decisión.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta **`make test`**

Instala el paquete:

```
make install
```

6.23.2. Contenido de Perl

Programas instalados: a2p, c2ph, cpan, dprofpp, enc2xs, find2perl, h2ph, h2xs, instmodsh, libnetcfg, perl, perl5.8.8 (enlace a perl), perlbug, perlcc, perldoc, perlivp, piconv, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, prove, psed (enlace a s2p), pstruct (enlace a c2ph), s2p, splain y xsubpp
Librerías instaladas: Varios cientos que no podemos listar aquí

Descripciones cortas

a2p Traduce de awk a Perl.

c2ph Vuelca estructuras C similares a las generadas por **cc -g -S**.

cpan Interactua con la red inteligente de archivos de Perl (Comprehensive Perl Archive Network, CPAN) desde la línea de comandos.

dprofpp Muestra datos de perfiles Perl.

enc2xs Construye una extensión Perl para el módulo Encode, a partir de cualquier Mapa de Caracteres Unicode o Ficheros de Codificación Tcl.

find2perl Traduce comandos **find** a código Perl.

h2ph Convierte ficheros de cabecera `.h` de C en ficheros de cabecera `.ph` de Perl.

h2xs Convierte ficheros de cabecera `.h` de C en extensiones de Perl.

instmodsh Guión para examinar los módulos de Perl instalados que incluso puede crear un paquete a partir de un módulo instalado.

libnetcfg Puede usarse para configurar `libnet`.

perl Combina algunas de las mejores características de C, **sed**, **awk** y **sh** en un único y poderoso lenguaje.

perl5.8.8 Enlace duro a **perl**.

perlbug Genera informes de errores sobre Perl o sobre los módulos incorporados y los envía por correo.

perlcc Genera ejecutables a partir de programas Perl.

perldoc Muestra una parte de la documentación en formato pod que se incluye en el árbol de instalación de Perl o en un guión de Perl.

perlivp El Procedimiento de Verificación de la Instalación de Perl. Puede usarse para verificar que Perl y sus librerías se han instalado correctamente.

piconv La versión Perl del convertidor de codificación de caracteres **iconv**.

pl2pm Es una herramienta que ayuda a convertir ficheros `.pl` de Perl4 en módulos `.pm` de Perl5.

pod2html Convierte ficheros de formato pod a formato HTML.

pod2latex Convierte ficheros de formato pod a formato LaTeX.

pod2man Convierte datos pod en entradas formateadas *roff.

pod2text Convierte datos pod en texto formateado ASCII.

pod2usage Muestra mensajes de uso a partir de documentos pod incluidos en ficheros.

podchecker Comprueba la sintaxis de los ficheros de documentación en formato pod.

podselect Muestra las secciones elegidas de la documentación pod.

prove Herramienta en línea de comandos para ejecutar pruevas contra el módulo Test::Harness.

psed Una versión Perl del editor de flujo **sed**.

pstruct Vuelca estructuras C similares a las generadas por **cc -g -S**.

s2p Traduce guiones de **sed** a Perl.

splain Se usa para forzar diagnósticos de avisos exhaustivos en Perl.

xsubpp Convierte el código XS de Perl en código C.

6.24. Readline-5.2

El paquete Readline contiene un conjunto de librerías que ofrecen edición de la línea de comandos y capacidades de historial.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 10.2 MB

6.24.1. Instalación de Readline

Una reinstalación Readline provacaría que las antiguas librerías fuesen movidas a <nombre_librería>.old. Aunque normalmente esto no es un problema, en algunos casos puede sacar a la luz un fallo de enlazado de **ldconfig**. Esto puede evitarse ejecutando las dos siguientes sustituciones:

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

Readline contiene un fallo en el manejo de caracteres multibyte que puede causar cálculos incorrectos de la pantalla. Corrige este problema aplicando aplicando el siguiente parche procedente de su desarrollador:

```
patch -Np1 -i ../readline-5.2-fixes-4.patch
```

Prepara Readline para su compilación:

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

Compila el paquete:

```
make SHLIB_LIBS=-lncurses
```

Significado de la opción de make:

`SHLIB_LIBS=-lncurses`
Esta opción fuerza a Readline a enlazarse contra la libería `libncurses` (en realidad, contra `libncursesw`).

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Mueve las librerías estáticas a una ubicación más correcta:

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

Ahora elimina los ficheros `.so` del directorio `/lib` y reenlazalos a `/usr/lib`:

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

6.24.2. Contenido de Readline

Librerías instaladas: libhistory.{a,so} y libreadline.{a,so}

Descripciones cortas

`libhistory` Proporciona una interfaz de usuario consistente para la rellamada de líneas de historial.

`libreadline` Asiste en la consistencia de la interfaz de usuario entre programas discrecionales que necesitan suministrar una interfaz de línea de comandos.

6.25. Zlib-1.2.3

El paquete Zlib contiene rutinas de compresión y descompresión usadas por algunos programas.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 3.1 MB

6.25.1. Instalación de Zlib

Nota

Se sabe que Zlib construye incorrectamente sus librerías si en el entorno se ha especificado un `CFLAGS`. Si estás usando tu propia variable `CFLAGS`, asegúrate de añadirle la directiva *`-fPIC`* durante el siguiente comando de configuración, y elimínala posteriormente.

Prepara Zlib para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala la librería compartida:

```
make install
```

El comando anterior instalo un fichero `.so` en `/lib`. Eliminalo y reenlazalo a `/usr/lib`:

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

Construye la librería estática:

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

Para obtener de nuevo los resultados de las pruebas, ejecuta: **`make check`**.

Instala la librería estática:

```
make install
```

Corrige los permisos de la librería estática:

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

6.25.2. Contenido de Zlib

Librerías instaladas: libz.{a,so}

Descripción corta

`libz` Contiene funciones de compresión y descompresión usadas por algunos programas.

6.26. Autoconf-2.61

El paquete Autoconf contiene programas para generar guiones del intérprete de comandos que pueden configurar automáticamente el código fuente.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 8.1 MB

6.26.1. Instalación de Autoconf

Prepara Autoconf para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**. Esto tarda bastante tiempo, unos 3 SBUs. Además, se ignoran 6 pruebas que necesitan Automake. Para una mayor cobertura de las pruebas puedes volver a probar Autoconf tras instalar Automake

Instala el paquete:

```
make install
```

6.26.2. Contenido de Autoconf

Programas instalados: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames

Descripciones cortas

autoconf Genera guiones del intérprete de comandos que automáticamente configuran paquetes de código fuente, adaptándolos a muchas clases de sistemas tipo UNIX. Los guiones de configuración que genera son independientes, para ejecutarlos no es necesario el programa **autoconf**.

autoheader Es una herramienta para crear plantillas de declaraciones #define de C, utilizadas por el guión configure.

autom4te Es un envoltorio para el procesador de macros M4.

autoreconf Ejecuta automáticamente, y en el orden correcto, **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize** y **libtoolize** para ahorrar tiempo cuando se hacen cambios en las plantillas de **autoconf** y **automake**.

autoscan Ayuda a crear un fichero `configure.in` para un paquete de software. Analiza los ficheros fuente en un árbol de directorios buscando problemas comunes de portabilidad y crea un fichero `configure.scan` que sirve como versión preliminar del fichero `configure.in` para dicho paquete.

autoupdate Modifica un fichero `configure.in` que todavía llame a las macros de **autoconf** por sus antiguos nombres para que utilice los nombre de macro actuales.

ifnames Ayuda a escribir ficheros `configure.in` para un paquete de software. Escribe los identificadores que el paquete usa en condicionales del preprocesador de C. Si un paquete está preparado para tener cierta portabilidad, este programa ayuda a determinar lo que **configure** necesita comprobar. Puede corregir ciertas carencias en un fichero `configure.in` generado por **autoscan**.

6.27. Automake-1.10

El paquete Automake contiene programas para generar Makefiles que se utilizan con Autoconf.

Tiempo estimado de construcción:	less than 0.1 SBU
Espacio requerido en disco:	7.9 MB

6.27.1. Instalación de Automake

Prepara Automake para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**. Esto tarda bastante tiempo, unos 10 SBUs.

Instala el paquete:

```
make install
```

6.27.2. Contenido de Automake

Programas instalados:	acinstall, aclocal, aclocal-1.10, automake, automake-1.10, compile, config.guess, config.sub, depcomp, elisp-comp, install-sh, mdate-sh, missing, mkinstalldirs, py-compile, symlink-tree e ylwrap

Descripciones cortas

acinstall	Guión que instala ficheros M4 con estilo aclocal.
aclocal	Genera ficheros `aclocal.m4` basados en el contenido de ficheros `configure.in`.
aclocal-1.10	Enlace duro a **aclocal**.
automake	Herramienta para generar automáticamente los `Makefile.in` a partir de ficheros `Makefile.am`. Para crear todos los ficheros `Makefile.in` para un paquete, ejecuta este programa en el directorio de más alto nivel. Mediante la exploración de los `configure.in` automáticamente encuentra cada `Makefile.am` apropiado y genera el correspondiente `Makefile.in`.
automake-1.10	Enlace duro a **automake**.
compile	Un envoltorio (wrapper) para compiladores.
config.guess	Guión que intenta averiguar el triplete canónico para la construcción, anfitrión o arquitectura destino dada.
config.sub	Guión con subrutinas para la validación de configuraciones.
depcomp	Guión para compilar un programa que, aparte de la salida deseada, también genera información sobre las dependencias.
elisp-comp	Compila en octetos código Lisp de Emacs.

install-sh	Guión que instala un programa, guión o fichero de datos.
mdate-sh	Guión que imprime la fecha de modificación de un fichero o directorio.
missing	Guión que actúa como sustituto común de programas GNU no encontrados durante una instalación.
mkinstalldirs	Guión que genera una árbol de directorios.
py-compile	Compila un programa Python.
symlink-tree	Guión para crear un árbol de enlaces simbólicos de un árbol de directorios.
ylwrap	Un envoltorio para **lex** y **yacc**.

6.28. Bash-3.2

El paquete Bash contiene la “Bourne-Again SHell”.

Tiempo estimado de construcción:	0.4 SBU
Espacio requerido en disco:	25.8 MB

6.28.1. Instalación de Bash

Si descargaste el paquete con la documentación de Bash y deseas instalar la documentación HTML, ejecuta los siguientes comandos:

```
tar -xvf ../bash-doc-3.2.tar.gz
sed -i "s|htmldir = @htmldir@|htmldir = /usr/share/doc/bash-3.2|" \
    Makefile.in
```

Aplica correcciones para varios fallos descubiertos dede la publicación inicial de Bash-3.2:

```
patch -Np1 -i ../bash-3.2-fixes-6.patch
```

Prepara Bash para su compilación:

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

Significado de la opción de configure:

`--with-installed-readline`
: Esta opción le indica a Bash que utilice la librería `readline` que se encuentra en el sistema, en vez de utilizar su propia versión de Readline.

Compila el paquete:

```
make
```

Si decides no ejecutar el banco de pruebas, salta hasta “Instala el paquete”.

Para preparar las pruebas, asegurate de que se usarán los ajuste de locale de nuestro entorno y de que el usuario `nobody` puede leer el dispositivo de entrada estandar y puede escribir en el árbol de las fuentes:

```
sed -i 's/LANG/LC_ALL/' tests/intl.tests
sed -i 's@tests@& </dev/tty@' tests/run-test
chown -Rv nobody ./
```

Ejecuta las pruebas como usuario `nobody`:

```
su-tools nobody -s /bin/bash -c "make tests"
```

Instala el paquete:

```
make install
```

Lanza el programa **bash** recién compilado (sustituyendo al que estabas ejecutando hasta ahora):

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

Nota

Los parámetros utilizados hacen del proceso **bash** un intérprete interactivo de ingreso y continúa desactivando su tabla interna de rutas para que los nuevos programas sean encontrados a medida que estén disponibles.

6.28.2. Contenido de Bash

Programas instalados: bash, bashbug y sh (enlace a bash)

Descripciones cortas

bash Un intérprete de comandos ampliamente usado. Realiza muchos tipos de expansiones y sustituciones en una línea de comandos dada antes de ejecutarla, lo que hace de este intérprete una herramienta poderosa.

bashbug Un guión que ayuda al usuario en la composición y envío de informes de errores relacionados con **bash**, usando un formato estándar.

sh Enlace simbólico al programa **bash**. Cuando se invoca como **sh**, **bash** intenta imitar el comportamiento de las versiones antiguas de **sh** lo mejor posible, mientras que también cumple los estándares POSIX.

6.29. Bzip2-1.0.4

El paquete Bzip2 contiene programas para comprimir y descomprimir ficheros. Comprimir ficheros de texto con **bzip2** proporciona un mayor porcentaje de compresión que el tradicional **gzip**.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 5.3 MB

6.29.1. Instalación de Bzip2

Aplica un parche para instalar la documentación de este paquete:

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

Prepara Bzip2 para su compilación:

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

Significado del parámetro de make:

tooldir=/usr
Esto provocará que Bzip2 sea construido usando un fichero `Makefile` diferente, en este caso el fichero `Makefile-libbz2_so`, el cual crea una librería dinámica `libbz2.so` y enlaza las utilidades de Bzip2 con ella.

Compila el paquete y comprueba los resultados:

```
make
```

Instala los programas:

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

Instala el binario dinámico **bzip2** en el directorio `/bin`, crea algunos enlaces simbólicos necesarios y haz limpieza:

```
cp -v bzip2-shared /bin/bzip2
cp -av libbz2.so* /lib
ln -sv ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so
rm -v /usr/bin/{bunzip2,bzcat,bzip2}
ln -sv bzip2 /bin/bunzip2
ln -sv bzip2 /bin/bzcat
```

6.29.2. Contenido de Bzip2

Programas instalados: bunzip2 (enlace a bzip2), bzcat (enlace a bzip2), bzcmp (enlace a bzdiff), bzdiff, bzegrep (enlace a bzgrep), bzfgrep (enlace a bzgrep), bzgrep, bzip2, bzip2recover, bzless (enlace a bzmore) y bzmore
Librerías instaladas: libbz2.{a,so}

Descripciones cortas

bunzip2 Descomprime ficheros que han sido comprimidos con **bzip2**.

bzcat Descomprime hacia la salida estándar.

bzcmp Ejecuta **cmp** sobre ficheros comprimidos con **bzip2**.

bzdiff Ejecuta **diff** sobre ficheros comprimidos con **bzip2**.

bzgrep Ejecuta **grep** sobre ficheros comprimidos con **bzip2**.

bzegrep Ejecuta **egrep** sobre ficheros comprimidos con **bzip2**.

bzfgrep Ejecuta **fgrep** sobre ficheros comprimidos con **bzip2**.

bzip2 Comprime ficheros usando el algoritmo de compresión de texto por ordenación de bloques Burrows-Wheeler con codificación Huffman. La compresión es, en general, considerablemente superior a la obtenida por otros compresores más convencionales basados en el algoritmo "Lempel-Ziv", como **gzip**.

bzip2recover Intenta recuperar datos de ficheros comprimidos dañados.

bzless Ejecuta **less** sobre ficheros comprimidos con **bzip2**.

bzmore Ejecuta **more** sobre ficheros comprimidos con **bzip2**.

`libbz2` La librería que implementa la compresión sin pérdidas por ordenación de bloques, usando el algoritmo de Burrows-Wheeler.

6.30. Diffutils-2.8.1

El paquete Diffutils contiene programas que muestran las diferencias entre ficheros o directorios.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 6.3 MB

6.30.1. Instalación de Diffutils

POSIX requiere que el comando **diff** trate los espacios en blanco de acuerdo con la locale actual. El siguiente parche corrige dicho incumplimiento:

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

El parche anteríor provocará que el sistema de construcción de Diffutils intente reconstruir la página de manual `diff.1` usando el programa **help2man**, que no está disponible. El resultado es una página de manual de **diff** ilegible. Podemos evitar esto actualizando la marca de tiempo del fichero `man/diff.1`:

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

Prepara Diffutils para su compilación:

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

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.30.2. Contenido de Diffutils

Programas instalados: cmp, diff, diff3 y sdiff

Descripciones cortas

cmp Compara dos ficheros e informa en dónde o en qué bytes difieren.

diff Compara dos ficheros o directorios e informa qué líneas de los ficheros difieren.

diff3 Compara tres ficheros línea a línea.

sdiff Mezcla dos ficheros y muestra los resultados interactivamente.

6.31. File-4.21

El paquete File contiene una utilidad para determinar el tipo de los ficheros.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	7.9 MB

6.31.1. Instalación de File

Prepara File para su compilación:

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

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.31.2. Contenido de File

Programa instalado:	file
Librerías instaladas:	libmagic.{a,so}

Descripciones cortas

file	Intenta clasificar los ficheros indicados. Lo hace realizando varias pruebas: pruebas de sistemas de ficheros, pruebas de números mágicos y pruebas de lenguajes.
`libmagic`	Contiene rutinas para reconocimiento de números mágicos, usados por el programa **file**.

6.32. Findutils-4.2.31

El paquete Findutils contiene programas para encontrar ficheros. Se suministran estos programas para hacer búsquedas recursivas en un árbol de directorios, y para crear, mantener y consultar una base de datos (más rápida que la búsqueda recursiva, pero imprecisa si la base de datos no se ha actualizado recientemente).

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 13.6 MB

6.32.1. Instalación de Findutils

Prepara Findutils para su compilación:

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

Significado de la opción de configure:

`--localstatedir`
Esta opción cambia la localización de la base de datos de **locate** para que se encuentre en `/var/lib/locate`, que cumple el FHS.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Algunos de los giones del paquete LFS-Bootscripts dependen de **find**. Como `/usr` puede no estar disponible en las primeras fases del arranque, este programa debe estar en la partición raíz. También debe corregirse una ruta explicta en el guión **updatedb**.

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

6.32.2. Contenido de Findutils

Programas instalados: bigram, code, find, frcode, locate, updatedb y xargs

Descripciones cortas

bigram Se usaba originalmente para generar bases de datos de **locate**.

code Se usaba originalmente para generar bases de datos de **locate**. Es el antecesor de **frcode**.

find Busca en los árboles de directorios indicados los ficheros que cumplan el criterio especificado.

frcode Es llamado por **updatedb** para comprimir la lista de nombres de ficheros. Utiliza "front-compression", que reduce el tamaño de la base de datos en un factor de 4 o 5.

locate Busca en una base de datos de nombres de ficheros y muestra los nombres que contienen la cadena indicada o cumplen un patrón dado.

updatedb Actualiza la base de datos de **locate**. Explora por completo el sistema de ficheros (incluidos otros sistemas de ficheros que se encuentren montados, a no ser que se le indique lo contrario) e inserta todos los nombres de ficheros que encuentre en la base de datos.

xargs Puede usarse para aplicar un comando a una lista de ficheros.

6.33. Flex-2.5.33

El paquete Flex contiene una utilidad para generar programas que reconocen patrones de texto.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	8.4 MB

6.33.1. Instalación de Flex

Prepara Flex para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Ciertos paquetes esperan encontrar la librería `lex` en el directorio `/usr/lib`. Crea un enlace simbólico para solventar esto:

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

Algunos programas aún no conocen **flex** e intentan encontrar a su predecesor **lex**. Para complacer a estos programas, crea un guión envoltorio de nombre `lex` que llame a **flex** en modo de emulación lex:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Begin /usr/bin/lex

exec /usr/bin/flex -l "$@"

# End /usr/bin/lex
EOF
chmod -v 755 /usr/bin/lex
```

6.33.2. Contenido de Flex

Programas instalados:	flex y lex
Librería instalada:	libfl.a

Descripciones cortas

flex	Una herramienta para generar programas capaces de reconocer patrones de texto. Su versatilidad permite establecer las reglas de búsqueda, erradicando la necesidad de desarrollar un programa especializado.
lex	Guión que ejecuta **flex** en el modo de emulación de **lex**.

`libfl.a` La librería `flex`.

6.34. GRUB-0.97

El paquete GRUB contiene el GRand Unified Bootloader (Gran Gestor de Arranque Unificado).

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 10.2 MB

6.34.1. Instalación de GRUB

Se sabe que este programa se comporta mal si se cambian sus parámetros de optimización (incluyendo las opciones *-march* y *-mcpu*). Si tienes definida cualquier variable de entorno que altere las optimizaciones por defecto, como CFLAGS o CXXFLAGS, desactívala cuando construyas GRUB.

Comienza aplicando el siguiente parche para permitir una mejor detección de los dispositivos, corregir algunos problemas con GCC 4.x y proporcionar un mejor soporte para algunos controladores de disco SATA:

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

Prepara GRUB para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **make check**.

Instala el paquete:

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

Sustituye i386-pc por el directorio apropiado para tu hardware.

El directorio i386-pc contiene también una serie de ficheros *stage1_5 para diferentes sistemas de ficheros. Mira los disponibles y copia el apropiado al directorio /boot/grub. La mayoría copiareis el fichero e2fs_stage1_5 y/o reiserfs_stage1_5.

6.34.2. Contenido de GRUB

Programas instalados: grub, grub-install, grub-md5-crypt, grub-set-default, grub-terminfo y mbchk

Descripciones cortas

grub El intérprete de comandos del GRand Unified Bootloader (Gran Gestor de Arranque Unificado).

grub-install Instala GRUB en el dispositivo indicado.

grub-md5-crypt Encripta una contraseña en formato MD5.

grub-set-default Establece la entrada de arranque por defecto para GRUB

grub-terminfo	Genera un comando terminfo a partir de un nombre terminfo. Puede utilizarse si tienes un terminal poco común.
mbchk	Comprueba el formato de un núcleo multiarranque.

6.35. Gawk-3.1.5

El paquete Gawk contiene programas para manipular ficheros de texto.

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 18.2 MB

6.35.1. Instalación de Gawk

Bajo ciertas circustancias, Gawk-3.1.5 intenta liberar un fragmento de memoria que no fué ocupado. Dicho error lo corrige el siguiente parche:

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

Prepara Gawk para su compilación:

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

Debido a un fallo en el guión **configure**, Gawk falla al detectar ciertos aspectos del soporte para locales de Glibc. Este error provoca, por ejemplo, fallos en el banco de pruebas de Gettext. Evita este problema añadiendo las definiciones de macro ausentes en `config.h`:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.35.2. Contenido de Gawk

Programas instalados: awk (enlace a gawk), gawk, gawk-3.1.5, grcat, igawk, pgawk, pgawk-3.1.5 y pwcat

Descripciones cortas

awk Enlace a **gawk**

gawk Un programa para manipular ficheros de texto. Es la implementación GNU de **awk**.

gawk-3.1.5 Enlace duro a **gawk**.

grcat Vuelca la base de datos de grupos `/etc/group`.

igawk Otorga a **gawk** la capacidad de incluir ficheros.

pgawk Es la versión de **gawk** con soporte de perfiles.

pgawk-3.1.5 Enlace duro a **pgawk**.

pwcat Vuelca la base de datos de contraseñas `/etc/passwd`.

6.36. Gettext-0.16.1

El paquete Gettext contiene utilidades para la internacionalización y localización. Esto permite a los programas compilarse con Soporte de Lenguaje Nativo (NLS), lo que les permite mostrar mensajes en el idioma nativo del usuario.

Tiempo estimado de construcción: 1 SBU
Espacio requerido en disco: 65 MB

6.36.1. Instalación de Gettext

Prepara Gettext para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**. Esto tarda mucho tiempo, unos 5 SBUs.

Instala el paquete:

```
make install
```

6.36.2. Contenido de Gettext

Programas instalados: autopoint, config.charset, config.rpath, envsubst, gettext, gettext.sh, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin y xgettext
Librerías instaladas: libasprintf.{a,so}, libgettextlib.so, libgettextpo.{a,so} y libgettextsrc.so

Descripciones cortas

autopoint Copia los ficheros estándar de infraestructura de Gettext a las fuentes de un paquete.

config.charset Saca una tabla dependiente del sistema de los alias de codificación de los caracteres.

config.rpath Saca un grupo de variables dependientes del sistema, describiendo cómo fijar la ruta de búsqueda en tiempo de ejecución de las librerías compartidas en un ejecutable.

envsubst Sustituye variables de entorno en cadenas del formato del intérprete de comandos.

gettext Traduce un mensaje en lenguaje natural al lenguaje del usuario, buscando las traducciones en un catálogo de mensajes.

gettext.sh Sirve principalmente como librería de funciones del interprete de comandos para Gettext

gettextize Copia todos los ficheros estándar Gettext en el directorio indicado de un paquete, para iniciar su internacionalización

hostname Muestra el nombre en la red de un sistema en varios formatos.

msgattrib Filtra los mensajes de un catálogo de traducción de acuerdo con sus atributos, y manipula dichos atributos.

msgcat Concatena y mezcla los ficheros `.po` indicados.

msgcmp Compara dos ficheros `.po` para comprobar si ambos contienen el mismo conjunto de cadenas de identificadores de mensajes.

msgcomm Busca los mensajes que son comunes en los ficheros `.po` indicados.

msgconv Convierte un catálogo de traducción a una codificación de caracteres diferente.

msgen Crea un catálogo de traducción en inglés.

msgexec Aplica un comando a todas las traducciones de un catálogo de traducción.

msgfilter Aplica un filtro a todas las traducciones de un catálogo de traducción.

msgfmt Compila el binario de un catálogo de mensajes a partir de un catálogo de traducciones.

msggrep Extrae todos los mensajes de un catálogo de traducción que cumplan cierto criterio o pertenezcan a alguno de los ficheros fuente indicados.

msginit Crea un nuevo fichero `.po`, inicializando la información con valores procedentes del entorno del usuario.

msgmerge Combina dos traducciones directas en un único fichero.

msgunfmt Descompila catálogos de mensajes binarios en traducciones directas de texto.

msguniq Unifica las traducciones duplicadas en un catálogo de traducción.

ngettext Muestra traducciones en lenguaje nativo de un mensaje textual cuya forma gramatical depende de un número.

recode-sr-latin Recodifica texto en serbio de cirílico a latín.

xgettext Extrae las líneas de mensajes traducibles de los ficheros fuente indicados, para hacer la primera plantilla de traducción.

`libasprintf` Define la clase *autosprintf* que hace utilizable la salida formateada de las rutinas de C en programas C++, para usar con las cadenas <*string*> y los flujos <*iostream*>.

`libgettextlib` Una librería privada que contiene rutinas comunes utilizadas por diversos programas de Gettext. No es indicada para uso general.

`libgettextpo` Utilizada para escribir programas especializados que procesan ficheros `.po`. Esta librería se utiliza cuando las aplicaciones estándar incluidas con Gettext no son suficiente (como **msgcomm**, **msgcmp**, **msgattrib** y **msgen**).

`libgettextsrc` Una librería privada que contiene rutinas comunes utilizadas por diversos programas de Gettext. No es indicada para uso general.

6.37. Grep-2.5.1a

El paquete Grep contiene programas para buscar dentro de ficheros.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 4.8 MB

6.37.1. Instalación de Grep

El paquete Grep actual contiene muchos errores, sobre todo en el soporte de locales multibyte. RedHat corrige algunos de ellos con el siguiente parche:

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

Para poder realñizar las pruebas añadidas por dicho parche, deben cambiarse los permisos del fichero de pruebas:

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

Prepara Grep para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.37.2. Contenido de Grep

Programas instalados: egrep, fgrep y grep

Descripciones cortas

egrep Muestra las líneas que coincidan con una expresión regular extendida.

fgrep Muestra las líneas que coincidan con una lista de cadenas fijas.

grep Muestra las líneas que coincidan con una expresión regular básica.

6.38. Groff-1.18.1.4

El paquete Groff contiene programas para procesar y formatear texto.

Tiempo estimado de construcción: 0.4 SBU

Espacio requerido en disco: 39.2 MB

6.38.1. Instalación de Groff

Aplica el parche para añadir los dispositivos “ascii8” y “nippon” a Groff:

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

Nota

Estos dispositivos son usados por Man-DB cuando se formatean páginas de manual que no están en inglés y no están codificadas en ISO-8859-1. Actuamente no hay un parche usable para Groff-1.19.x que añada dicha funcionalidad.

Muchas fuentes de pantalla no incluyen las comillas simples y dobles de Unicode. Indicale a Groff que utilice en su lugar los equivalentes ASCII:

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

Groff espera que la variable de entorno PAGE contenga el valor por defecto para el tamaño de papel. Para los residentes en Estados Unidos, *PAGE=letter* es un valor adecuado. Para el resto, *PAGE=A4* puede ser más correcto. Aunque el tamaño del papel por defecto se configura durante la compilación, puede sobreescribirse posteriormente cambiando “A4” o “letter” en el fichero /etc/papersize.

Prepara Groff para su compilación:

```
PAGE=<paper_size> ./configure --prefix=/usr --enable-multibyte
```

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Algunos programas de documentación, como **xman**, no funcionarán correctamente sin los siguientes enlaces simbólicos.

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

6.38.2. Contenido de Groff

Programas instalados: addftinfo, afmtodit, eqn, eqn2graph, geqn (enlace a eqn), grn, grodvi, groff, groffer, grog, grolbp, grolj4, grops, grotty, gtbl (enlace a tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, pic2graph, post-grohtml, pre-grohtml, refer, soelim, tbl, tfmtodit y troff

Descripciones cortas

addftinfo Lee un fichero de fuentes troff y añade alguna información adicional sobre la métrica de la fuente, que es usada por el sistema **groff**.

afmtodit Crea un fichero de fuentes para usarlo con **groff** y **grops**.

eqn Compila las descripciones de las fórmulas embebidas en los ficheros de entrada troff a comandos que pueda entender **troff**.

eqn2graph Convierte una ecuación EQN en una imagen.

geqn Enlace a **eqn**

grn Un preprocesador **groff** para ficheros gremlin.

grodvi Un controlador para **groff** que genera formatos dvi de TeX.

groff Una interfaz para el sistema de formateado de documentos groff. Normalmente lanza el programa **troff** y un post-procesador apropiado para el dispositivo seleccionado.

groffer Muestra ficheros groff y páginas de manual en las X y en consola.

grog Lee ficheros y averigua cuál de las opciones `-e`, `-man`, `-me`, `-mm`, `-ms`, `-p`, `-s` y `-t` de **groff** se necesitan para imprimir los ficheros, y muestra el comando de **groff** incluyendo esas opciones.

grolbp Un controlador de **groff** para las impresoras Canon CAPSL (series LBP-4 y LBP-8 de impresoras láser)

grolj4 Un controlador para **groff** que produce salidas en el formato PCL5 adecuado para impresoras HP LaserJet 4.

grops Transforma la salida de GNU **troff** a PostScript.

grotty Transforma la salida de GNU **troff** en un formato adecuado para dispositivos tipo máquina de escribir.

gtbl Enlace a **tbl**.

hpftodit Crea un fichero de fuentes para usar con **groff -Tlj4** a partir de ficheros de marcas de fuentes métricas de HP.

indxbib Hace un índice inverso para la base de datos bibliográfica, un fichero específico para usarlo con **refer**, **lookbib** y **lkbib**.

lkbib Busca en las bases de datos bibliográficas referencias que contengan las claves especificadas y muestra cualquier referencia encontrada.

lookbib Muestra un aviso en la salida de error estándar (excepto si la entrada estándar no es un terminal), lee de la entrada estándar una línea conteniendo un grupo de palabras clave, busca en las bases de datos bibliográficas de un fichero especificado las referencias que contengan dichas claves, muestra cualquier referencia encontrada en la salida estándar y repite el proceso hasta el final de la entrada.

mmroff	Un preprocesador simple para **groff**.
neqn	Formatea ecuaciones para salida ASCII (Código Estándar Americano para Intercambio de Información).
nroff	Un guión que emula al comando **nroff** usando **groff**.
pfbtops	Transforma una fuente en formato `.pfb` de PostScript a ASCII.
pic	Compila descripciones de gráficos embebidos dentro de ficheros de entrada troff o TeX a comandos que puedan ser entendidos por TeX o **troff**.
pic2graph	Convierte un diagrama PIC en una imagen.
post-grohtml	Transforma la salida de GNU **troff** a HTML.
pre-grohtml	Transforma la salida de GNU **troff** a HTML.
refer	Copia el contenido de un fichero en la salida estándar, excepto que las líneas entre *.[* y *.]* son interpretadas como citas, y las líneas entre *.R1* y *.R2* son interpretadas como comandos sobre cómo deben ser procesadas las citas.
soelim	Lee ficheros y reemplaza líneas de la forma *fichero .so* por el contenido de *fichero*.
tbl	Compila descripciones de tablas embebidas dentro de ficheros de entrada troff a comandos que puedan ser entendidos por **troff**.
tfmtodit	Crea un fichero de fuentes para su uso con **groff -Tdvi**.
troff	Es altamente compatible con Unix **troff**. Normalmente debe ser invocado usando el comando **groff**, que también lanzará los preprocesadores y post procesadores en el orden correcto y con las opciones necesarias.

6.39. Gzip-1.3.12

El paquete Gzip contiene programas para comprimir y descomprimir ficheros.

Tiempo estimado de construcción:	less than 0.1 SBU
Espacio requerido en disco:	2.2 MB

6.39.1. Instalación de Gzip

La versión de la función “futimens” usada por Gzip es incompatible con la versión actual proporcionada por Glibc. Por tanto, renombra la función:

```
sed -i 's/futimens/gl_&/' gzip.c lib/utimens.{c,h}
```

Prepara Gzip para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Mueve algunos programas que no necesitan estar en el sistema de ficheros raíz:

```
mv -v /bin/{gzexe,uncompress,zcmp,zdiff,zegrep} /usr/bin
mv -v /bin/{zfgrep,zforce,zgrep,zless,zmore,znew} /usr/bin
```

6.39.2. Contenido de Gzip

Programas instalados:	gunzip, gzexe, gzip, uncompress, zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore y znew

Descripciones cortas

gunzip	Descomprime ficheros que hayan sido comprimidos con **gzip**.
gzexe	Crea ficheros ejecutables autodescomprimibles.
gzip	Comprime los ficheros indicados usando codificación Lempel-Ziv (LZ77).
uncompress	Descomprime ficheros comprimidos.
zcat	Descomprime en la salida estándar los ficheros indicados comprimidos con **gzip**.
zcmp	Ejecuta **cmp** sobre ficheros comprimidos.
zdiff	Ejecuta **diff** sobre ficheros comprimidos.
zegrep	Ejecuta **egrep** sobre ficheros comprimidos.

zfgrep Ejecuta **fgrep** sobre ficheros comprimidos.

zforce Fuerza la extensión `.gz` en todos los ficheros comprimidos para que **gzip** no los comprima dos veces. Esto puede ser útil para ficheros con el nombre truncado después de una transferencia de ficheros.

zgrep Ejecuta **grep** sobre ficheros comprimidos.

zless Ejecuta **less** sobre ficheros comprimidos.

zmore Ejecuta **more** sobre ficheros comprimidos.

znew Recomprime ficheros del formato de **compress** al formato de **gzip**, o sea, de `.Z` a `.gz`.

6.40. Inetutils-1.5

El paquete Inetutils contiene programas para trabajo básico en red.

Tiempo estimado de construcción:	0.2 SBU
Espacio requerido en disco:	8.9 MB

6.40.1. Instalación de Inetutils

No vamos a instalar todos los programas que vienen en Inetutils. Sin embargo, el sistema de construcción de Inetutils insistirá en instalar todas las páginas de manual. El siguiente parche corregirá esta situación:

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

Prepara Inetutils para su compilación:

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

Significado de las opciones de configure:

`--disable-ifconfig`
Esta opción evita que Inetutils instale **ifconfig**, que podría usarse para configurar interfaces de red. LFS utiliza **ip** de IPRoute2 para realizar esta tarea.

`--disable-logger`
Esta opción evita que Inetutils instale el programa **logger**, que sirve para que los guiones le pasen mensajes al Demonio de Registro de Eventos del Sistema. Hacemos esto porque luego Util-linux instalará una versión mejor.

`--disable-syslogd`
Esta opción evita que Inetutils instale el Demonio de Registro de Eventos del Sistema, que será instalado con el paquete Sysklogd.

`--disable-whois`
Esta opción desactiva la construcción del cliente **whois** de Inetutils, que está demasiado anticuado. En el libro BLFS hay instrucciones para un cliente **whois** mucho mejor.

`--disable-servers`
Esto desactiva la construcción de los diferentes servidores incluidos como parte del paquete Inetutils. Estos servidores no se consideran apropiados para un sistema LFS básico. Algunos son inseguros por naturaleza y sólo se consideran seguros en redes de confianza. Puedes encontrar más información en *http://www.linuxfromscratch.org/blfs/view/svn/basicnet/inetutils.html*. Ten en cuenta que para muchos de estos servidores hay disponibles sustitutos mejores.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Mueve el programa **ping** al lugar indicado por el FHS:

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

6.40.2. Contenido de Inetutils

Programas instalados: ftp, ping, ping6, rcp, rlogin, rsh, talk, telnet y tftp

Descripciones cortas

ftp El programa para transferencia de ficheros de ARPANET.

ping Envía paquetes de petición de eco e informa cuánto tardan las respuestas.

ping6 Versión de **ping** para redes IPv6.

rcp Copia ficheros de forma remota.

rlogin Realiza entradas remotas a un sistema.

rsh Ejecuta un intérprete de comandos remoto.

talk Permite hablar con otro usuario.

telnet Una interfaz de usuario para el protocolo TELNET.

tftp Un programa para la transferencia trivial de ficheros.

6.41. IPRoute2-2.6.20-070313

El paquete IPRoute2 contiene programas para el trabajo básico y avanzado en redes basadas en IPV4.

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 4.8 MB

6.41.1. Instalación de IPRoute2

La instalación de dos páginas de manual está rota, generando enlaces simbólicos muertos. Corrige esto con el siguiente comando:

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

Compila el paquete:

```
make SBINDIR=/sbin
```

Significado de la opción de make:

`SBINDIR=/sbin`
Esto asegura que los binarios de IPRoute2 se instalarán en `/sbin`. Esta es la localización correcta según el FHS, pues algunos de los binarios de IPRoute2 se utilizan en los guiones de arranque.

Este paquete no incluye un banco de pruebas.

Instala el paquete:

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

El binario **arpd** se enlaza contra las librerías Berkeley DB que residen en `/usr` y utiliza una base de datos en `/var/lib/arpd/arpd.db`. Por tanto, según el FHS, debe estar en `/usr/sbin`. Muevelo allí:

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

6.41.2. Contenido de IPRoute2

Programas instalados: arpd, ctstat (enlace a lnstat), genl, ifcfg, ifstat, ip, lnstat, nstat, routef, routel, rtacct, rtmon, rtpr, rtstat (enlace a lnstat), ss y tc

Descripciones cortas

arpd Demonio ARP a nivel de usuario, útil en redes realmente grandes en las que la implementación ARP del núcleo es insufuciente, o cuando se configura un "honeypot".

ctstat Utilidad para el estado de la conexión.

genl

ifcfg Un guión del intérprete de comandos que actúa como envoltorio para el comando **ip**.

ifstat Muestra las estadísticas de las interfaces, incluida la cantidad de paquetes enviados y recibidos por la interfaz.

ip El ejecutable principal. Tiene diferentes funciones:

ip link <*dispositivo*> permite a los usuarios ver el estado del dispositivo y hacer cambios.
ip addr permite a los usuarios ver las direcciones y sus propiedades, añadir nuevas direcciones y borrar las antiguas.
ip neighbor permite a los usuarios ver los enlaces de vecindad, añadir nuevas entradas de vecindad y borrar las antiguas.
ip rule permite a los usuarios ver las políticas de enrutado y cambiarlas.
ip route permite a los usuarios ver las tablas de enrutado y cambiar las reglas de las tablas.
ip tunnel permite a los usuarios ver los túneles IP y sus propiedades, y cambiarlos.
ip maddr permite a los usuarios ver las direcciones multienlace y sus propiedades, y cambiarlas.
ip mroute permite a los usuarios establecer, cambiar o borrar el enrutado multienlace.
ip monitor permite a los usuarios monitorizar continuamente el estado de los dispositivos, direcciones y rutas.

lnstat Proporciona estadísticas de redes Linux. Es un sustituto generalista y con características más completas para el antiguo programa **rtstat**.

nstat Muestra las estadísticas de la red.

routef Un componente de **ip route**. Este es para refrescar las tablas de enrutado.

routel Un componente de **ip route**. Este es para listar las tablas de enrutado.

rtacct Muestra el contenido de `/proc/net/rt_acct`.

rtmon Utilidad para la monitorización de rutas.

rtpr Convierte la salida de **ip -o** a un formato legible

rtstat Utilidad para el estado de rutas.

ss Similar al comando **netstat**. Muestra las conexiones activas.

tc Ejecutable para el control del tráfico. Este es para las implementaciones Quality Of Service (QOS, Calidad de Servicio) y Class Of Service (COS, Clase de Servicio).
tc qdisc permite a los usuarios establecer la disciplina de colas.
tc class permite a los usuarios establecer clases basadas en la planificación de las disciplinas de colas.
tc estimator permite a los usuarios hacer una estimacón del flujo de red en una red.
tc filter permite a los usuarios establecer el filtrado de paquetes QOS/COS.
tc policy permite a los usuarios establecer las políticas QOS/COS.

6.42. Kbd-1.12

El paquete Kbd contiene ficheros de mapas de teclado y utilidades para el teclado.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 12.3 MB

6.42.1. Instalación de Kbd

El comportamiento de las teclas Retroceso y Borrado no es homogeneo en los diferentes maspas de teclado del paquete Kbd. El siguiente parche corrige este problema para los mapas de teclado i386:

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

Tras parchear, la tecla de Retroceso genera el carácter con código 127, y la tecla de borrado genera una secuencia de escape bien conocida.

Parchea Kbd para corregir un error en **setfont** que aparece cuando se compila con GCC-4.2.1:

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

Prepara Kbd para su compilación:

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

Significado de la opción de configure:

`--datadir=/lib/kbd`
Esta opción pone los datos de esquema de teclado en un directorio que siempre se encontrará en la partición raíz. en vez de en `/usr/share/kbd`.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Nota

Para algunos idiomas (por ejemplo, bieloruso) el paquete Kdb no proporciona un mapa del teclado útil (el mapa de teclado “by” incluido asume la codificación ISO-8859-5, mientras que normalmente se utiliza CP1251).Los usuarios de dichos idiomas deberán descargar aparte mapas de teclado funcionales.

Algunos de los guiones del paquete LFS-Bootscripts dependen de **kbd_mode**, **openvt** y **setfont**. Como `/usr` puede no estar disponible en las primeras fases del arranque, estos binarios deben estar en la partición raíz:

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

6.42.2. Contenido de Kbd

Programas instalados: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (enlace a psfxtable), psfgettable (enlace a psfxtable), psfstriptable (enlace a psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setmetamode, showconsolefont, showkey, unicode_start y unicode_stop

Descripciones cortas

chvt Cambia la terminal virtual que aparece en primer plano.

deallocvt Desasigna las terminales virtuales no usadas.

dumpkeys Vuelca las tablas de traducción del teclado.

fgconsole Muestra el número del terminal virtual activo.

getkeycodes Muestra la tabla de correspondencias de código de exploración (scan code) a código de teclas del núcleo.

kbd_mode Muestra o establece el modo del teclado.

kbdrate Establece la repetición y retardo del teclado.

loadkeys Carga las tablas de traducción del teclado.

loadunimap Carga la tabla de correspondencia de unicode a fuente del núcleo.

mapscrn Un programa obsoleto que carga una tabla de correspondencia de caracteres de salida, definida por el usuario, en el controlador de la consola. Esto lo hace ahora **setfont**.

openvt Comienza un programa en un nuevo terminal virtual (VT).

psf* Son un grupo de herramientas para manejar tablas de caracteres Unicode para fuentes de consola.

resizecons Cambia la idea del núcleo sobre el tamaño de la consola.

setfont Permite cambiar las fuentes EGA y VGA de la consola.

setkeycodes Carga las entradas de la tabla de correspondencia de código de exploración (scan code) a código de teclas del núcleo. Es últil si el teclado tiene teclas inusuales.

setleds Establece los LEDs y las opciones del teclado. Mucha gente encuentra útil tener el bloqueo numérico (Num Lock) activado por defecto.

setmetamode Define cómo se manejan las teclas meta del teclado.

showconsolefont Muestra la fuente de pantalla EGA/VGA actual de la consola.

showkey Muestra los códigos de exploración, códigos de tecla y códigos ASCII de las teclas presionadas en el teclado.

unicode_start Pone el teclado y la consola en modo UNICODE. No uses este programa a menos que tu mapa de teclado sea para codoficación ISO-8859-1. Para otras codificaciones, esta utilidad produce resultados incorrectos.

unicode_stop Revierte el teclado y la consola del modo UNICODE.

6.43. Less-406

El paquete Less contiene un visor de ficheros de texto.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 2.8 MB

6.43.1. Instalación de Less

Prepara Less para su compilación:

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

Significado de la opción de configure:

`--sysconfdir=/etc`
Esta opción le indica al programa creado por el paquete que busque en `/etc` sus ficheros de configuración.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.43.2. Contenido de Less

Programas instalados: less, lessecho y lesskey

Descripciones cortas

less Un visor de ficheros o paginador. Muestra el contenido de un fichero con la posibilidad de recorrerlo, hacer búsquedas o saltar a marcas.

lessecho Necesario para expandir meta-caracteres, como * y *?*, en los nombres de ficheros en sistemas Unix.

lesskey Usado para especificar los códigos de teclas usados por **less**.

6.44. Make-3.81

El paquete Make contiene un programa para compilar paquetes.

Tiempo estimado de construcción:	0.1 SBU
Espacio requerido en disco:	9.6 MB

6.44.1. Instalación de Make

Prepara Make para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.44.2. Contenido de Make

Programa instalado: make

Descripción corta

make Determina automáticamente qué partes de un paquete necesitan ser (re)compiladas y lanza los comandos para hacerlo.

6.45. Man-DB-2.4.4

El paquete Man-DB contiene programas para encontrar y visualizar páginas de manual.

Tiempo estimado de construcción:	0.2 SBU
Espacio requerido en disco:	9 MB

6.45.1. Instalación de Man-DB

Es necesario hacer cuatro ajustes a las fuentes de Man-DB.

El primero cambia la localización de las páginas de manual traducidas que vienen con Man-DB, para poder acceder a ellas tanto con locales tradicionales como UTF-8:

```
mv man/de{_DE.88591,}
mv man/es{_ES.88591,}
mv man/it{_IT.88591,}
mv man/ja{_JP.eucJP,}
sed -i 's,\*_\*,??,' man/Makefile.in
```

El segundo cambio es una sustitución **sed** para borrar las líneas “/usr/man” y “/usr/local/man” del fichero `man_db.conf` para evitar resultados duplicados cuando se utilizan programas como **whatis**:

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

El tercer cambio tiene en cuenta programas que Man-DB debería ser capaz de encontrar en tiempo de ejecución, pero que no han sido instalados aún:

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

El programa **col** es parte del paquete Util-linux, **lynx** es un navegador web en modo texto (mira en BLFS las instrucciones de instalación), **vgrind** convierte fuentes de programas a entrada Groff, y **grap** es útil para la inclusión de gráficos en documentos Groff. Los programas **vgrind** y **grap** no son necesarios normalmente para ver páginas de manual. No son parte de LFS o BLFS, pero deberías ser capaz de instalarlos por ti mismo tras finalizar el LFS si así lo deseas.

Por último, parchea las fuentes para corregir errores en la salida si la página de manual es abortada prematuramente mediante la pulsación de la tecla 'q':

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

Prepara Man-DB para su compilación:

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

Significado de las opciones de configure:

`--enable-mb-groff`
Esto le indica al programa **man** que utilice los dispositivos “ascii8” y “nippon” de Groff para formatear páginas de manual que no estén en ISO-8859-1.

`--disable-setuid`
Esto desactiva que el programa **man** se instale con setuid al usuario `man`.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Algunos paquetes proporcionan páginas de manual en UTF-8 que esta versión de **man** no puede mostrar. El siguiente guión permitirá que algunas de ellas sean convertidas a las codificaciones esperadas que se listan abajo. Man-DB espera que las páginas de manual estén en las codificaciones de la tabla, y las convertirá según sea necesario a la codificación real de la locale cuando las muestre, por lo que las mostrará tanto en ocales UTF-8 como en locales tradicionales. Debido a que este guión está pensado para un uso limitado durante la construcción del sistema, para datos públicos, no nos precupamos por la comprobación de errores ni usamos ficheros temporales con nombres no predecibles:

```
cat >> convert-mans << "EOF"
#!/bin/sh -e
FROM="$1"
TO="$2"
shift ; shift
while [ $# -gt 0 ]
do
        FILE="$1"
        shift
        iconv -f "$FROM" -t "$TO" "$FILE" >.tmp.iconv
        mv .tmp.iconv "$FILE"
done
EOF
install -m755 convert-mans  /usr/bin
```

Información adicional sobre la compresión de páginas de manual e info se puede encontrar en el libro BLFS en *http://www.linuxfromscratch.org/blfs/view/stable/postlfs/compressdoc.html*.

6.45.2. Páginas de manual no inglesas en LFS

Las distribuciones Linux tienen diferentes políticas en cuanto a la codificación de caractéres en los que las páginas de manual se almacenan en el sistema de ficheros. Por ejemplo, RedHat almacena todas las páginas de manual en UTF-8, mientras que Debian utiliza la codificación específica del idioma (generalmente en 8 bits). Esto produce incompatibilidad entre los paquetes con paginas de manual diseñadas por diferentes distribuciones.

LFS utiliza la misma política que Debian. Esta fué elegida porque Man-DB no entiende las páginas almacenadas en UTF-8. Y, para nuestros propósitos, Man-DB es preferible a Man pues funciona sin configuraciones adicionales para ninguna locale. Por último, y a día de hoy, no existe una implementación con funcionalidad completa de la política de RedHat. Se sabe que el **groff** the RedHat formatea mal el texto.

Tabla 6.1. Codificación asumida de caracteres de las páginas de manual

Idioma (código)	Codificación
Danés (da)	ISO-8859-1
Alemán (de)	ISO-8859-1
Inglés (en)	ISO-8859-1
Español (es)	ISO-8859-1
Finlandés (fi)	ISO-8859-1
Francés (fr)	ISO-8859-1
Irlandés (ga)	ISO-8859-1
Gallego (gl)	ISO-8859-1
Indonesio (id)	ISO-8859-1
Islandés (is)	ISO-8859-1
Italiano (it)	ISO-8859-1
Holandés (nl)	ISO-8859-1
Noruego (no)	ISO-8859-1
Portugés (pt)	ISO-8859-1
Sueco (sv)	ISO-8859-1
Checo (cs)	ISO-8859-2
Croata (hr)	ISO-8859-2
Húngaro (hu)	ISO-8859-2
Japonés (ja)	EUC-JP
Coreano (ko)	EUC-KR
Polaco (pl)	ISO-8859-2
Ruso (ru)	KOI8-R
Slovaco (sk)	ISO-8859-2
Turko (tr)	ISO-8859-9

Nota

Las páginas de manual en idiomas que no se encuentren en la lista no estás soportadas. Noruego no funciona ahora debido a la transición de la locale no_NO a nb_NO, y el Coreano no es funcional debido a que el parche de Groff es incompleto.

Si el desarrollador distribuye las páginas de manual con la misma codificación que Man-DB espera, estas pueden copiarse a `/usr/share/man/<código del idioma>`. Por ejemplo, las páginas de manual en Francés (*http://ccb.club.fr/man/man-fr-1.58.0.tar.bz2*) pueden instalarse con el siguiente comando:

```
mkdir -p /usr/share/man/fr
cp -rv man? /usr/share/man/fr
```

Si el desarrollador distribuye las páginas de manual en UTF-8 (por ejemplo, “para RedHat”) en vez de en la codificación listada en la tabla anterior, tendrán que convertirse de UTF-8 a la codificación listada antes de instalarlas. Esto puede hacerse con **convert-mans**, por ejemplo, las páginas de manual en Español (*http://ditec.um.es/~piernas/manpages-es/man-pages-es-1.55.tar.bz2*) pueden instalarse con los siguientes comandos:

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

Nota

La necesidad de excluir el fichero `man7/iso_8859-7.7` del proceso de conversión debido a que ya está en ISO-8859-1es por un error en el empaquetado de man-pages-es-1.55. Futuras versiones no deberían necesitar este apaño.

6.45.3. Contenido de Man-DB

Programas instalados: accessdb, apropos, catman, convert-mans, lexgrog, man, mandb, manpath, whatis y zsoelim

Descripciones cortas

accessdb Vuelca el contenido de la base de datos de **whatis** a formato legible.

apropos Búsca en la base de datos de **whatis** y muestra las descripciones cortas de los comandos del sistema que contienen la cadena dada.

catman Crea o actualiza las páginas de manual preformateadas

convert-mans Reformatea páginas de manual para que Man-DB pueda leerlas.

lexgrog Muestra información sumarizada en una línea sobre la página de manual dada.

man Formatea y muestra la página de manual solicitada.

mandb Crea o actualiza la base de datos de **whatis**.

manpath Muestra el contenido de $MANPATH o (si $MANPATH no está definido) la ruta de búsqueda disponible basada en los ajuste de man.conf y el entorno del usuario.

whatis Búsca en la base de datos de **whatis** y muestra las descripciones cortas de los comandos del sistema que contienen la palabra clave dada como palabra separada.

zsoelim Lee ficheros y reemplaza líneas del tipo *.so fichero* por el contenido del *file* citado.

6.46. Mktemp-1.5

El paquete Mktemp contiene programas usados para crear ficheros temporales seguros en guiones de intérpretes de comandos.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 0.4 MB

6.46.1. Instalación de Mktemp

Muchos programas todavía usan el anticuado programa **tempfile**, que tiene una funcionalidad similar a **mktemp**. Parchea Mktemp para incluir un envoltorio **tempfile**:

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

Prepara Mktemp para su compilación:

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

Significado de la opción de configure:

`--with-libc`
Esto hace que el programa **mktemp** utilice las funciones *mkstemp* y *mkdtemp* de la librería C del sistema en lugar de su propia implementación.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
make install-tempfile
```

6.46.2. Contenido de Mktemp

Programas instalados: mktemp y tempfile

Descripciones cortas

mktemp Crea ficheros temporales de forma segura. Es usado en guiones.

tempfile Crea ficheros temporales de una forma menos segura que **mktemp**. Se instala por retro-compatibilidad.

6.47. Module-Init-Tools-3.2.2

El paquete Module-Init-Tools contiene programas para manejar módulos del núcleo en núcleos Linux con versión mayor o igual a 2.5.47.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 7 MB

6.47.1. Instalación de Module-Init-Tools

Primero corrige un problema potencial cuando los módulos son especificados usando expresiones regulares:

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

Ejecuta los siguientes comandos para efectuar las pruebas (advierte que el comando **make distclean** es necesario para limpiar el árbol de las fuentes, pues las fuentes son recompiladas como parte del proceso de pruebas):

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

Prepara Module-Init-Tools para su compilación:

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

Compila el paquete:

```
make
```

Instala el paquete:

```
make INSTALL=install install
```

Significado del parámetro de make:

`INSTALL=install`
Normalmente, **make install** no instalará los binários si estos ya existen. Esta opción modifica dicho comportamiento invocando a **install** en vez de usar el guión envoltorio utilizado por defecto.

6.47.2. Contenido de Module-Init-Tools

Programas instalados: depmod, generate-modprobe.conf, insmod, insmod.static, lsmod, modinfo, modprobe y rmmod

Descripciones cortas

depmod Crea un fichero de dependencias basándose en los símbolos que encuentra en el conjunto existente de módulos del núcleo. A este fichero lo usa **modprobe** para cargar automáticamente los módulos necesarios.

generate-modprobe.conf Crea un fichero modprobe.conf a partir de una configuración de módulos 2.2 o 2.4 existente.

insmod	Instala un módulo dentro del núcleo en ejecución.
insmod.static	Una versión de **insmod** compilada estáticamente.
lsmod	Muestra todos los módulos cargados.
modinfo	Examina un fichero objeto asociado con un módulo del núcleo y muestra la información que pueda encontrar.
modprobe	Usa un fichero de dependencias, creado por **depmod**, para cargar automáticamente los módulos necesarios.
rmmod	Descarga módulos del núcleo en ejecución.

6.48. Patch-2.5.4

El paquete Patch contiene un programa para modificar o crear ficheros mediante la aplicación de un fichero “parche” creado normalmente con el programa **diff**.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 1.6 MB

6.48.1. Instalación de Patch

Prepara Patch para su compilación:

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

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.48.2. Contenido de Patch

Programa instalado: patch

Descripción corta

patch Modifica ficheros según lo indicado en un fichero parche. Normalmente un parche es una lista de diferencias creada por el programa **diff**. Al aplicar estas diferencias a los ficheros originales, **patch** crea las versiones parcheadas.

6.49. Psmisc-22.5

El paquete Psmisc contiene programas para mostrar información sobre procesos en ejecución.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 2.2 MB

6.49.1. Instalación de Psmisc

Prepara Psmisc para su compilación:

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

Significado de la opción de configure:

`--exec-prefix=""`
Esto asegura que los binarios de Psmisc se instalen en `/bin` en lugar de `/usr/bin`. Este es el lugar correcto según el FHS, pues algunos de los binarios de Psmisc son usados por el paquete LFS-Bootscripts.

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

No hay razón para que los programas **pstree** y **pstree.x11** residan en `/bin`. Por tanto los moveremos a `/usr/bin`:

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

El programa **pidof** de Psmisc no se instala por defecto. Normalmente esto no es ningún problema, ya que más tarde instalaremos el paquete Sysvinit, el cual nos facilita una versión mejor del programa **pidof**. Pero si no vas a usar Sysvinit, debes completar la instalación de Psmisc creando el siguiente enlace simbólico:

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

6.49.2. Contenido de Psmisc

Programas instalados: fuser, killall, oldfuser, peekfd, pstree y pstree.x11 (enlace a pstree)

Descripciones cortas

fuser Muestra los números de identificación (PID) de los procesos que usan los ficheros o sistemas de ficheros especificados.

killall Mata procesos por su nombre. Envía una señal a todos los procesos que ejecutan alguno de los comandos especificados.

oldfuser Muestra los números de identificación (PIDs) de los procesos que utilizan los ficheros o sistemas de ficheros dados.

peekfd	Vigila los descriptores de ficheros de un proceso, dado su PID
pstree	Muestra los procesos en ejecución en forma de árbol.
pstree.x11	Es igual que **pstree** excepto que espera confirmación antes de salir.

6.50. Shadow-4.0.18.1

El paquete Shadow contiene programas para manejar contraseñas de forma segura.

Tiempo estimado de construcción:	0.3 SBU
Espacio requerido en disco:	20.7 MB

6.50.1. Instalación de Shadow

Nota

Si deseas forzar el uso de contraseñas fuertes, consulta *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/cracklib.html* para instalar Cracklib antes de construir Shadow. Entonces añade `--with-libcrack` al siguiente comando **configure**.

Corrige un error en los programas **useradd** y **usermod** que les impide aceptar nombres de grupo en vez de números ID de grupo al usar la opción `-g`:

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

Prepara Shadow para su compilación:

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

Significado de la opción de configure:

`--without-selinux`
El soporte para selinux está activado por defecto, pero selinux no se construye en el sistema LFS base. El guión **configure** fallará si no se utiliza esta opción.

Suprime la instalación del programa **groups** y sus páginas de manual, pues Coreutils proporciona una versión mejor:

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

Desactiva la instalación de las páginas de manual en chino y coreano, pues Man-DB no puede formatearlas correctamente:

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

Shadow incluye otras páginas de manual en codificación UTF-8. Man-DB podrá mostrarlas en las codificaciones recomendadas usando el guión **convert-mans** que instalamos anteriormente:

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

for i in cs hu pl; do
    convert-mans UTF-8 ISO-8859-2 man/${i}/*.?
done

convert-mans UTF-8 EUC-JP man/ja/*.?
convert-mans UTF-8 KOI8-R man/ru/*.?
convert-mans UTF-8 ISO-8859-9 man/tr/*.?
```

En vez de usar el método por defecto, *crypt*, utiliza el método de encriptación de contraseñas *MD5*, que es más seguro y además permite contraseñas de más de 8 caracteres. También es necesario cambiar la obsoleta localización `/var/spool/mail`, que Shadow utiliza por defecto para los buzones de los usuarios, a `/var/mail`, que es la localización usada hoy en día:

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

Nota

Si construyes Shadow con soporte para Cracklib, ejecuta lo siguiente:

```
sed -i 's@DICTPATH.*@DICTPATH\t/lib/cracklib/pw_dict@' \
    etc/login.defs
```

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

Mueve un programa mal ubicado a su lugar correcto:

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

Mueve las librerías de Shadow a un lugar más apropiado:

```
mv -v /lib/libshadow.*a /usr/lib
rm -v /lib/libshadow.so
ln -sfv ../../lib/libshadow.so.0 /usr/lib/libshadow.so
```

6.50.2. Configuración de Shadow

Este paquete contiene utilidades para añadir, modificar o eliminar usuarios y grupos, establecer y cambiar sus contraseñas y otras tareas administrativas. Puedes encontrar una completa explicación de lo que significa *password shadowing* (ocultación de contraseñas) en el fichero `doc/HOWTO` dentro del árbol de las fuentes. Hay una cosa que debes recordar si decides usar soporte para Shadow: los programas que necesiten verificar contraseñas (administradores de sesión, programas FTP, demonios pop3, etc) necesitarán ser compatibles con Shadow: esto es, necesitan ser capaces de trabajar con contraseñas ocultas.

Para habilitar las contraseñas ocultas, ejecuta el siguiente comando:

```
pwconv
```

Para habilitar las contraseñas de grupo ocultas, ejecuta:

```
grpconv
```

La configuración base de Shadow para la utilidad **useradd** no es la correcta para un sistema LFS. Utiliza los siguientes comandos para cambiar el directorio personal por defecto para nuevos usuarios y evitar la creación de sus ficheros de almacen de correo:

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

6.50.3. Establecer la contraseña de root

Elige una contraseña para el usuario *root* y establécela mediante:

```
passwd root
```

6.50.4. Contenido de Shadow

Programas instalados: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, lastlog, login, logoutd, newgrp, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (enlace a newgrp), su, useradd, userdel, usermod, vigr (enlace a vipw) y vipw

Librerías instaladas: libshadow.{a,so}

Descripciones cortas

chage Se usa para cambiar el número máximo de días entre cambios obligatorios de contraseña.

chfn Se usa para cambiar el nombre completo de un usuario y otra información.

chgpasswd Utilizado para actualizar lotes de contraseñas de grupos.

chpasswd Utilizado para actualizar lotes de contraseñas de usuarios.

chsh Cambia el intérprete de comandos por defecto que se ejecuta cuando el usuario entra al sistema.

expiry Comprueba y refuerza la política actual de expiración de contraseñas.

faillog Sirve para examinar el contenido del registro de ingresos fallidos al sistema, establecer un máximo de fallos para bloquear una cuenta de usuario y reiniciar el contador de fallos.

gpasswd Se usa para agregar y eliminar miembros y administradores a los grupos.

groupadd Crea un nuevo grupo con el nombre especificado.

groupdel Borra el grupo con el nombre especificado.

groupmems Permite a un usuario administrar la lista de miembros de su propio grupo sin necesidad de privilegios de superusuario.

groupmod Modifica el nombre o el identificador (GID) de un grupo especificado.

grpck Verifica la integridad de los ficheros de grupos, `/etc/group` y `/etc/gshadow`.

grpconv Crea o actualiza el fichero de grupos ocultos a partir de un fichero de grupos normal.

grpunconv Actualiza `/etc/group` a partir de `/etc/gshadow`, borrando este último.

lastlog Muestra el último acceso de cada usuario o de un usuario especificado.

login Lo utiliza el sistema para permitir el ingreso de un usuario.

logoutd Es un demonio que refuerza las restricciones de ingreso en base a horas y puertos de acceso.

newgrp Se usa para cambiar el identificador de grupo (GID) actual durante una sesión de acceso.

newusers Crea o actualiza un grupo de cuentas de usuario de una sola vez.

nologin Muestra un mensaje sobre que una cuenta no está disponible. Diseñado para usarse como interprete de comandos por defecto para cuentas que han sido desactivadas.

passwd Se utiliza para cambiar la contraseña de la cuenta de un usuario o grupo.

pwck Verifica la integridad de los ficheros de contraseñas, `/etc/passwd` y `/etc/shadow`.

pwconv Crea o actualiza el fichero de contraseñas ocultas a partir de un fichero de contraseñas normal.

pwunconv Actualiza `/etc/passwd` a partir de `/etc/shadow`, borrando este último.

sg Ejecuta un comando dado estableciendo el GID del usuario al del grupo indicado.

su Ejecuta un intérprete de comandos sustituyendo los identificadores de usuario y grupo.

useradd Crea un nuevo usuario con el nombre especificado o actualiza la información por defecto de un nuevo usuario.

userdel Borra la cuenta de usuario indicada.

usermod Modifica el nombre, identificador (UID), intérprete de comandos, grupo inicial, directorio personal, etc, del usuario indicado.

vigr Edita los ficheros `/etc/group` o `/etc/gshadow`.

vipw Edita los ficheros `/etc/passwd` o `/etc/shadow`.

`libshadow` Contiene funciones usadas por la mayoría de los programas de este paquete.

6.51. Sysklogd-1.5

El paquete Sysklogd contiene programas para registrar los mensajes del sistema, como aquellos generados por el núcleo cuando sucede algo inusual.

Tiempo estimado de construcción: less than 0.1 SBU
Espacio requerido en disco: 0.6 MB

6.51.1. Instalación de Sysklogd

Compila el paquete:

```
make
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make install
```

6.51.2. Configuración de Sysklogd

Crea un nuevo fichero `/etc/syslog.conf` ejecutando lo siguiente:

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

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

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

6.51.3. Contenido de Sysklogd

Programas instalados: klogd y syslogd

Descripciones cortas

klogd Un demonio del sistema que intercepta y registra los mensajes del núcleo.

syslogd Registra los mensajes que los programas del sistema ofrecen. Cada mensaje registrado contine al menos una marca de tiempo y un nombre de máquina y, normalmente, también el nombre del programa, pero depende de cómo de verboso se le pide al demonio de registro que sea.

6.52. Sysvinit-2.86

El paquete Sysvinit contiene programas para controlar el arranque, ejecución y cierre del sistema.

Tiempo estimado de construcción: less than 0.1 SBU

Espacio requerido en disco: 1 MB

6.52.1. Instalación de Sysvinit

Cuando se cambia de nivel de ejecución (por ejemplo cuando apagamos el sistema) el programa **init** envía las señales de finalización a aquellos procesos que él mismo inició y que no deben estar en ejecución en el nuevo nivel. Mientras lo hace, **init** muestra mensajes del tipo “Sending processes the TERM signal” (Enviando la señal TERM a los procesos), que parece indicar que se está enviando dicha señal a todos los procesos que hay en ejecución. Para evitar esta confusión, puedes modificar las fuentes para que ese mensaje diga en su lugar “Sending processes configured via /etc/inittab the TERM signal” (Enviando la señal TERM a los procesos configurados en /etc/inittab):

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

Compila el paquete:

```
make -C src
```

Este paquete no incluye un banco de pruebas.

Instala el paquete:

```
make -C src install
```

6.52.2. Configuración de Sysvinit

Crea un nuevo fichero `/etc/inittab` ejecutando lo siguiente:

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

id:3:initdefault:

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

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

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

su:S016:once:/sbin/sulogin

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

# End /etc/inittab
EOF
```

6.52.3. Contenido de Sysvinit

Programas instalados: bootlogd, halt, init, killall5, last, lastb (enlace a last), mesg, mountpoint, pidof (enlace a killall5), poweroff (enlace a halt), reboot (enlace a halt), runlevel, shutdown, sulogin, telinit (enlace a init), utmpdump y wall

Descripciones cortas

bootlogd Registra los mensajes de arranque en un fichero.

halt Suele invocar a **shutdown** con la opción *-h*, excepto cuando el sistema ya se encuentra en el nivel de ejecución 0, en cuyo caso le indica al núcleo que apague el sistema. Anota en `/var/log/wtmp` que el sistema se va a cerrar.

init El primer proceso que se inicia cuando el núcleo ha inicializado el hardware, el cual toma el control sobre el arranque e inicia todos los procesos que se le han indicado.

killall5 Envía una señal a todos los procesos, excepto a los procesos de su propia sesión para que no mate el intérprete de comandos desde el que fue llamado.

last Muestra los últimos usuarios conectados (y desconectados), buscando hacia atrás en el fichero `/var/log/wtmp`. También muestra los inicios y paradas del sistema y los cambios de nivel de ejecución.

lastb Muestra los intentos fallidos de acceso al sistema, que se registran en `/var/log/btmp`.

mesg Controla si otros usuarios pueden o no enviar mensajes al terminal del usuario actual.

mountpoint Comprueba si el directorio es un punto de montaje.

pidof Muestra los identificadores de proceso (PIDs) de los programas especificados.

poweroff Le indica al núcleo que cierre el sistema y apague la máquina (ver **halt**).

reboot Le indica al núcleo que reinicie el sistema (ver **halt**).

runlevel Muestra los niveles de ejecución anterior y actual tal y como figura en el último registro de nivel de ejecución de `/var/run/utmp`.

shutdown Provoca el cierre del sistema de una forma segura, enviando señales a todos los procesos y notificando a todos los usuarios conectados.

sulogin Permite el ingreso de `root` al sistema. Suele ser invocado por **init** cuando el sistema entra en el modo monousuario.

telinit Le indica a **init** a qué nivel de ejecución debe cambiar.

utmpdump Muestra el contenido de un fichero de registro de accesos dado en un formato comprensible por el usuario.

wall Envía un mensaje a todos los usuarios conectados.

6.53. Tar-1.18

El paquete Tar contiene un programa de archivado.

Tiempo estimado de construcción:	0.3 SBU
Espacio requerido en disco:	19.9 MB

6.53.1. Instalación de Tar

Prepara Tar para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

6.53.2. Contenido de Tar

Programas instalados: rmt y tar

Descripciones cortas

rmt Manipula remotamente una unidad de cinta magnética mediante una comunicación de conexión entre procesos.

tar Crea, extrae ficheros y lista el contenido de un archivo, también conocido como paquete tar (tarball).

6.54. Texinfo-4.9

El paquete Texinfo contiene programas usados para leer, escribir y convertir páginas info.

Tiempo estimado de construcción: 0.2 SBU
Espacio requerido en disco: 16.6 MB

6.54.1. Instalación de Texinfo

El programa **info** asume que una cadena ocupa el mismo número de celdas de caracter en la pantalla que bytes en memoria y que se puede cortar una cade por cualquier parte, lo que no es cierto con locales basadas en UTF-8. El siguiente parche lo solventa al hacer que se muestren los mensajes en inglés cuando se utiliza una locale multibyte:

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

Texinfo permite a usuarios locales sobreescribir ficheros arbitrarios mediante un ataque de enlace simbólico sobre ficheros temporales. Aplica el siguiente parche para corregir esto:

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

Prepara Texinfo para su compilación:

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

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make check`**.

Instala el paquete:

```
make install
```

Opcionalmente, instala los componentes que pertenecen a una instalación de TeX:

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

Significado del parámetro de make:

`TEXMF=/usr/share/texmf`
La variable `TEXMF` del Makefile fija la ubicación de la raíz del árbol de TeX si, por ejemplo, más adelante se instala un paquete TeX.

El sistema de documentación Info utiliza un fichero de texto plano para almacenar su listado de entradas de menú. Este fichero se encuentra en `/usr/share/info/dir`. Desgraciadamente, debido a problemas ocasionales en los Makefile de diversos paquetes, en ocasiones puede quedarse desfasado con respecto a las páginas info realmente instaladas en el sistema. Si necesitas recrear el fichero `/usr/share/info/dir`, el siguiente comando opcional hará el trabajo:

```
cd /usr/share/info
rm dir
for f in *
do install-info $f dir 2>/dev/null
done
```

6.54.2. Contenido de Texinfo

Programas instalados: info, infokey, install-info, makeinfo, texi2dvi, texi2pdf y texindex

Descripciones cortas

info	Lee páginas info, que son similares a las páginas de manual, pero tienden a ser más profundos que una simple explicación de las opciones de un programa. Por ejemplo, compara **man bison** con **info bison**.
infokey	Compila un fichero fuente que contiene opciones de Info en un formato binario.
install-info	Se usa para instalar páginas info. Actualiza las entradas en el fichero índice de **info**.
makeinfo	Convierte documentos fuente Texinfo a páginas info, texto plano, o HTML.
texi2dvi	Formatea un documento Texinfo, convirtiéndolo en un fichero independiente del dispositivo que puede ser impreso.
texi2pdf	Se usa para formatear un documento Texinfo como fichero Portable Document Format (PDF).
texindex	Se usa para ordenar ficheros índice de Texinfo.

6.55. Udev-113

El paquete Udev contiene programas para la creación dinámica de nodos de dispositivos.

Tiempo estimado de construcción: 0.1 SBU
Espacio requerido en disco: 5.8 MB

6.55.1. Instalación de Udev

El paquete udev-config contiene ficheros específicos de LFS para configurar Udev. Desempaquetalo dentro del directorio de las fuentes de Udev:

```
tar -xvf ../udev-config-20070731.tar.bz2
```

Crea algunos dispositivos y directorios que Udev no puede manejar debido a que son necesarios al principio del proceso de arranque:

```
install -dv /lib/{firmware,udev/devices/{pts,shm}}
mknod -m0666 /lib/udev/devices/null c 1 3
ln -sv /proc/self/fd /lib/udev/devices/fd
ln -sv /proc/self/fd/0 /lib/udev/devices/stdin
ln -sv /proc/self/fd/1 /lib/udev/devices/stdout
ln -sv /proc/self/fd/2 /lib/udev/devices/stderr
ln -sv /proc/kcore /lib/udev/devices/core
```

Compila el paquete:

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

Significado de la opción de make:

EXTRAS=...
Esto construye los binarios que pueden ayudar en la escritura de reglas Udev personalizadas.

Para comprobar los resultados, ejecuta: **make test**.

Advierte que el banco de pruebas de Udev generará numerosos mensajes en los ficheros de registro del sistema anfitrión. Estos son inofensivos y pueden ignorarse.

Instala el paquete:

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

Significado del parámetro de make:

DESTDIR=/
Esto evita que el proceso de instalación de Udev mate cualquier proceso **udevd** que pueda estar ejecutandose en el sistema anfitrión.

Para que funcione correctamente se ha de configurar Udev, pues este sólo instala unos pocos configuración por defecto. Instala primero los ficheros de reglas usados comúnmente proporcionados por Udev:

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

Instala ahora los ficheros de reglas específicos de LFS:

```
cd udev-config-20070731
make install
```

Instala la documentación que esplica las reglas específicas de LFS:

```
make install-doc
```

Instala la documentación que esplica las reglas comúnes proporcionadas por Udev:

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

Instala la documentación que esplica cómo crear reglas Udev personalizadas:

```
cd ..
install -m644 -v docs/writing_udev_rules/index.html \
    /usr/share/doc/udev-113/index.html
```

6.55.2. Contenido de Udev

Programas instalados: ata_id, cdrom_id, create_floppy_devices, edd_id, firmware.sh, path_id, scsi_id, udevcontrol, udevd, udevinfo, udevmonitor, udevsettle, udevtest, udevtrigger, usb_id, vol_id, write_cd_rules y write_net_rules
Librería instalada: libvolume_id
Directorio instalado: /etc/udev

Descripciones cortas

ata_id Proporciona a Udev una cadena única e información adicional (uuid, label) para un controlador ATA.

cdrom_id Proporciona a Udev las capacidades de un cotrolador CD-ROM o DVD-ROM.

create_floppy_devices Crea todos los tipos posibles de dispositivos de disquete basados en el tipo CMOS.

edd_id Proporciona a Udev el ID EDD de un controlador BIOS de disco.

firmware.sh Carga firmware en dispositivos.

path_id Proporciona la ruta hardware mas corta posible a un dispositivo.

scsi_id Proporciona a Udev un identificados SCSI único basado en los datos devueltos tras el envio de un comando SCSI INQUIRY al dispositivo especificado.

udevcontrol Configura ena serie de opciones para el demonio **udevd** en ejecución, como el nivel de registro.

udevd Un demonio que escucha uevents en un conector de red, crea dispositivos y ejecuta los programas externos configurados en respuesta a dichos uevents.

udevinfo Permite a los usuarios consultar en la base de datos de Udev información sobre cualquier dispositivo actualmente presente en el sistema. También proporciona un método para consultar cualquier dispositivo en el árbol `sysfs` ayudando a crear reglas Udev.

udevmonitor Muestra el evento recibido del núcleo y el entorno que Udev envía tras el proceso de reglas.

udevsettle	Vigila la colo de eventos Udev y sale si todos los uevents actuales han sido manejados.
udevtest	Simula un uevent para el dispositivo dado y muestra el nombre del nodo que el **udevd** real podría crear, o el nombre de la interfaz de red renombrada.
udevtrigger	Dispara los uevents de dispositivos del núcleo para que sean repetidos.
usb_id	Proporciona a Udev información sobre dispositivos USB.
vol_id	Proporciona a Udev la etiqueta y el uuid de un sistema de ficheros.
write_cd_rules	Guión que genera reglas Udev proporcionando nombres estáticos para dispositivos ópticos (ver también Sección 7.12, “Crear enlaces simbólicos persnalizados a los dispositivos”).
write_net_rules	Guión que genera reglas Udev proporcionando nombres estáticos para interfaces de red (ver también Sección 7.13, “Configuración del guión network”).
libvolume_id	
`/etc/udev`	Contiene ficheros de configuración de Udev, permisos de dispositivos y reglas para la denominación de dispositivos.

6.56. Util-linux-2.12r

El paquete Util-linux contiene una miscelánea de utilidades. Entre otras hay utilidades para manejar sistemas de ficheros, consolas, particiones y mensajes.

Tiempo estimado de construcción:	0.2 SBU
Espacio requerido en disco:	17.2 MB

6.56.1. Notas sobre la conformidad con el estándar FHS

El estándar FHS recomienda que usemos `/var/lib/hwclock` para la ubicación del fichero `adjtime`, en lugar del habitual `/etc`. Para hacer que **hwclock** sea conforme a FHS, ejecuta lo siguiente:

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

6.56.2. Instalación de Util-linux

Util-linux falla al compilarse contra las nuevas versiones de las cabeceras del núcleo Linux. Los siguientes parches corrigen dichos problemas:

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

Prepara Util-linux para su compilación:

```
./configure
```

Compila el paquete:

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

Significado de los parámetros de make:

HAVE_KILL=yes
: Esto evita que el programa **kill** (que ya ha sido instalado por Procps) sea construido e instalado de nuevo.

HAVE_SLN=yes
: Esto evita que el programa **sln** (un **ln** enlazado estáticamente, ya instalado por Glibc) se vuelva a construir e instalar.

Este paquete no incluye un banco de pruebas.

Instala el paquete:

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

6.56.3. Contenido de Util-linux

Programas instalados: agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, flock, fsck.cramfs, fsck.minix, getopt, hexdump, hwclock, ipcrm, ipcs, isosize, line, logger, look, losetup, mcookie, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, namei, pg, pivot_root, ramsize (link to rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (link to rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (link to swapon), swapon, tailf, tunelp, ul, umount, vidmode (link to rdev), whereis y write

Descripciones cortas

agetty Abre un puerto de terminal, espera la introducción de un nombre de usuario e invoca al comando **login**.

arch Muestra la arquitectura de la máquina.

blockdev Permite llamar a los controles de entrada/salida (ioctls) de los dispositivos de bloque desde la línea de comandos.

cal Muestra un calendario simple.

cfdisk Se usa para manipular la tabla de particiones del dispositivo indicado.

chkdupexe Encuentra ejecutables duplicados.

col Elimina avances de línea inversos.

colcrt Filtra la salida de **nroff** para terminales a los que les faltan ciertas características como el sobrerefresco o semilíneas.

colrm Elimina las columnas indicadas.

column Formatea un fichero a múltiples columnas.

ctrlaltdel Establece la función de la combinación de teclas Ctrl+Alt+Del para un reinicio duro o blando.

cytune Ajusta los parámetros de los controladores de línea serie para tarjetas Cyclades.

ddate Muestra la fecha Discordante, o convierte las fechas Gregorianas en fechas Discordantes.

dmesg Muestra los mensajes de arranque del núcleo.

elvtune Puede usarse para afinar el rendimiento y la interactividad de un dispositivo de bloque.

fdformat Formatea un disquete a bajo nivel.

fdisk Se usa para manipular la tabla de particiones del dispositivo indicado.

flock Adquiere un fichero de bloqueo y ejecuta un comando con el bloqueo activado.

fsck.cramfs Realiza una comprobación de consistencia sobre el sistema de ficheros Cramfs del dispositivo indicado

fsck.minix Realiza una comprobación de consistencia en sistemas de ficheros Minix.

getopt Analiza opciones de la línea de comandos indicada.

hexdump Muestra un fichero en hexadecimal o en otro formato.

hwclock Se usa para leer o ajustar el reloj del ordenador, también llamado RTC (Reloj en Tiempo Real) o reloj BIOS (Sistema Básico de Entrada/Salida).

ipcrm Elimina el recurso IPC (Comunicación Entre Procesos) especificado.

ipcs Facilita información sobre el estado IPC.

isosize Muestra el tamaño de un sistema de ficheros iso9660.

line Copia una única línea.

logger Crea entradas en el registro del sistema.

look Muestra líneas que comienzan con una cadena dada.

losetup Activa y controla los dispositivos de bucle (loop).

mcookie Genera galletas mágicas (magic cookies, números hexadecimales aleatorios de 128 bits) **para xauth**.

mkfs Construye un sistema de ficheros en un dispositivo (normalmente una partición del disco duro).

mkfs.bfs Crea un sistema de ficheros bfs de SCO (Operaciones Santa Cruz).

mkfs.cramfs Crea un sistema de ficheros Cramfs.

mkfs.minix Crea un sistema de ficheros Minix.

mkswap Inicializa el dispositivo o fichero indicado para usarlo como área de intercambio (swap).

more Un filtro para paginar texto pantalla a pantalla.

mount Monta el sistema de ficheros de un dispositivo dado en el directorio indicado del árbol de ficheros del sistema.

namei Muestra los enlaces simbólicos en la ruta de nombres indicada.

pg Muestra un fichero de texto a pantalla completa.

pivot_root Hace que el sistema de ficheros indicado sea el raíz del proceso actual.

ramsize Se usa para establecer el tamaño del disco RAM en una imagen de arranque.

raw Utilizado para enlazar un dispositivo Linux de caracteres directo a un dispositivo de bloque.

rdev Muestra y establece el dispositivo raíz, entre otras cosas, en una imagen de arranque.

readprofile Lee la información sobre perfiles del núcleo.

rename Renombra ficheros, sustituyendo la cadena indicada con otra.

renice Altera la prioridad de los procesos en ejecución.

rev Invierte el orden de las líneas de un fichero.

rootflags Se usa para establecer las opciones de partición raíz en una imagen de arranque.

script Hace un guión a partir de una sesión de terminal.

setfdprm Establece los parámetros facilitados por el usuario para los disquetes.

setsid Lanza programas en una nueva sesión.

setterm Establece los parámetros del terminal.

sfdisk Un manipulador de la tabla de particiones del disco.

swapoff Desactiva los dispositivos y ficheros de paginación e intercambio.

swapon Activa los dispositivos y ficheros de paginación e intercambio y lista los dispositivos y ficheros en uso.

tailf Observa el crecimiento de un fichero de registro. Muestra las últimas 10 líneas de un fichero y continua mostrando cualquier nueva entrada en el fichero cuando es creada.

tunelp	Ajusta los parámetros de la línea de impresión.
ul	Un filtro para traducir marcas de texto a la secuencia de escape que indica subrayado para el terminal en uso.
umount	Desmonta un sistema de ficheros del árbol de ficheros del sistema.
vidmode	Establece el modo de vídeo en una imagen de arranque.
whereis	Localiza el binario, las fuentes y la página del manual de un comando.
write	Envía un mensaje a otro usuario *si* ese usuario no ha desactivado dichos mensajes.

6.57. Vim-7.1

El paquete Vim contiene un poderoso editor de texto.

Tiempo estimado de construcción:	0.4 SBU
Espacio requerido en disco:	47.4 MB

Alternativas a Vim

Si prefieres otro editor en vez de Vim, como Emacs, Joe, o Nano, mira en *http://www.linuxfromscratch.org/blfs/view/stable/postlfs/editors.html* las instrucciones de instalación sugeridas.

6.57.1. Instalación de Vim

Primero, desempaqueta en el mismo directorio tanto `vim-7.1.tar.bz2` como (opcionalmente) `vim-7.1-lang.tar.gz`.

Aplica un parche que corrige diversos problemas encontrados y corregidos por los desarrolladores desde la liberalización inicial de Vim-7.1:

```
patch -Np1 -i ../vim-7.1-fixes-2.patch
```

Esta versión de Vim instala las páginas de manual traducidas en directorios en los que Man-DB no las buscará. Parchea Vim para que instale sus páginas de manual en los directorios apropiados y permita a Man-DB transcodificar la página al formato deseado:

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

Por último, cambia la localización por defecto del fichero de configuración `vimrc` a `/etc`.

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

Prepara Vim para su compilación:

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

Significado de la opción de configure:

`--enable-multibyte`
Este parámetro añade a **vim** el soporte para la edición de ficheros codificados con caracteres multibyte. Esto es necesario si se utiliza un conjunto de caracteres multibyte. También permite editar ficheros creados inicialmente en distribuciones Linux como Fedora Core, que utilizan UTF-8 como conjunto de caracteres por defecto.

Compila el paquete:

```
make
```

Para comprobar los resultados, ejecuta: **`make test`**. Sin embargo, este banco de pruebas mostrará por pantalla un montón de datos binarios que pueden causar problemas con los ajustes del terminal actual. Esto puede evitarse redirigiendo la salida a un fichero de registro.

Instala el paquete

```
make install
```

Muchos usuarios están acostumbrados a usar **vi**, en vez de **vim**. Para permitirles ejecutar **vim** cuando teclean **vi**, crea enlaces simbólicos tanto para el binario como para la página de manual en los idiomas suministrados:

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

Por defecto, la documentación de Vim se instala en `/usr/share/vim`.El siguiente enlace permite que la documentación sea accesible mediante `/usr/share/doc/vim-7.1`, haciendolo consistente con la localización de la documentación del resto de paquetes:

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

Si vas a instalar un sistema X Window en tu sistema LFS, puede que sea necesario recompilar Vim después de instalar X. Vim incluye una bonita versión con interfaz gráfica que necesita X y algunas otras librerías instaladas. Para más información lee la documentación de Vim y la página de instalación de Vim en el libro BLFS, en *http://www.linuxfromscratch.org/blfs/view/svn/postlfs/editors.html#postlfs-editors-vim*.

6.57.2. Configuración de Vim

Por defecto, **vim** se ejecuta en modo no compatible con vi. Esto puede ser nuevo para los usuarios que han utilizado otros editores anteriormente. Se incluye a continuación la opción “nocompatible” para resaltar el hecho de que se va a usar este nuevo comportamiento. Esto también les recuerda a aquellos que quieran cambiar al modo “compatible” que debería ser la primera entrada en el fichero de configuración. Esto es necesario porque cambia otros ajustes, y las modificaciones deberían ir tras este ajuste. Crea un fichero de configuración por defecto de **vim** ejecutando lo siguiente:

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

set nocompatible
set backspace=2
syntax on
if (&term == "iterm") || (&term == "putty")
  set background=dark
endif

" End /etc/vimrc
EOF
```

La opción *`set nocompatible`* hace que **vim** se comporte de un modo (el modo por defecto) más útil que el modo compatible con vi. Elimina el “no” si quieres el antiguo comportamiento **vi**. La opción *`set backspace=2`* permite el retroceso en saltos de línea, autoindentación e inicio de inserción. La opción *syntax on* activa la coloración semántica de **vim**. Por último, el condicional *if* junto con *`set background=dark`* corrige lo que **vim** se imagina sobre el color de fondo de ciertos emuladores de terminal. Esto le da a la coloración semántica un mejor esquema de color para utilizarlo sobre el fondo negro de estos programas.

Se puede obtener información sobre las opciones disponibles ejecutando el siguiente comando:

```
vim -c ':options'
```

Nota

Por defecto, Vim instala ficheros de corrección ortográfica solamente para inglés. Para instalar los ficheros para tu idioma preferido, descarga los ficheros `*.spl`, y opcionalmente los `*.sug`, para tu idioma y codificación de caractéres desde *ftp://ftp.vim.org/pub/vim/runtime/spell/* y guardalos en `/usr/share/vim/vim71/spell/`

Para utilizar estos ficheros ortográficos es necesario configurar `/etc/vimrc`, por ejemplo:

```
set spelllang=en,es
set spell
```

Para más información, mira el README que se encuentra en la anterior URL.

6.57.3. Contenido de Vim

Programas instalados: efm_filter.pl, efm_perl.pl, ex (enlace a vim), less.sh, mve.awk, pltags.pl, ref, rview (enlace a vim), rvim (enlace a vim), shtags.pl, vi (enlace a vim), view (enlace a vim), vim, vim132, vim2html.pl, vimdiff (enlace a vim), vimm, vimspell.sh, vimtutor y xxd

Descripciones cortas

efm_filter.pl Un filtro para crear un fichero de error que puede ser leído por **vim**.

efm_perl.pl Formatea los mensajes de error del intérprete Perl para usarlos con el modo “quickfix” de **vim**.

ex Arranca **vim** en modo ex.

less.sh Un guión que arranca **vim** con less.vim.

mve.awk Procesa los errores de **vim**.

pltags.pl Crea un fichero de etiquetas para el código Perl, de modo que pueda usarse con **vim**.

ref Comprueba la ortografía de los argumentos.

rview Una versión restringida de **view**. No pueden ejecutarse comandos del intérprete de comandos y **view** no puede ser suspendido.

rvim Una versión restringida de **vim**. No pueden ejecutarse comandos del intérprete de comandos y **vim** no puede ser suspendido.

shtags.pl Genera un fichero de etiquetas para los guiones Perl.

vi Enlace a **vim**

view Arranca **vim** en modo de sólo lectura.

vim El editor.

vim132 Arranca **vim** con el terminal en modo de 132 columnas.

vim2html.pl Convierte la documentación de Vim a HTML.

vimdiff Edita dos o tres versiones de un fichero con **vim** y muestra las diferencias.

vimm Activa el modelo de entrada del buscador de DEC en un terminal remoto.

vimspell.sh	Comprueba la ortografía de un fichero y genera las sentencias de sintaxis necesarias para resaltar las palabras en **vim**. Este guión necesita el antiguo comando Unix **spell**, que no se incluye en el LFS ni en el BLFS.
vimtutor	Enseña las teclas y comandos básicos de **vim**.
xxd	Genera un volcado hexadecimal. También puede hacer lo contrario, por lo que puede usarse para parchear binarios.

6.58. Sobre los símbolos de depuración

La mayoría de los programas y librerías se compilan por defecto incluyendo los símbolos de depuración (con la opción *-g* de **gcc**). Esto significa que, cuando se depura un programa o librería que fue compilado incluyendo la información de depuración, el depurador no nos da sólo las direcciones de memoria, sino también los nombres de las rutinas y variables.

Sin embargo, la inclusión de estos símbolos de depuración agranda sustancialmente un programa o librería. Para tener una idea del espacio que ocupan estos símbolos, echa un vistazo a lo siguiente:

- Un binario **bash** con símbolos de depuración: 1200 KB
- Un binario **bash** sin símbolos de depuración: 480 KB
- Los ficheros de Glibc y GCC (`/lib` y `/usr/lib`) con símbolos de depuración: 87 MB
- Los ficheros de Glibc y GCC sin símbolos de depuración: 16 MB

Los tamaños pueden variar algo dependiendo del compilador y la librería C utilizadas, pero cuando comparamos programas con y sin símbolos de depuración, la diferencia generalmente está en una relación de entre 2 y 5.

Como muchas personas probablemente nunca usen un depurador en su sistema, eliminando estos símbolos se puede liberar una gran cantidad de espacio del disco. Para tu comodidad, la siguiente sección muestra cómo eliminar todos los símbolos de depuración de los programas y librerías. Puedes encontrar información sobre otras formas de optimizar tu sistema en la receta *http://www.linuxfromscratch.org/hints/downloads/files/optimization.txt*.

6.59. Eliminar los símbolos de nuevo.

Si no eres un programador y no planeas depurar el software de tu sistema, puedes reducir tu sistema en unos 90 MB eliminando los símbolos de depuración de los binarios y librerías. Este proceso no produce ningún otro inconveniente que no sea no poder depurar los programas nunca más.

La mayoría de la gente que usa el comando mencionado más adelante no experimenta ningún problema. Pero es fácil cometer un error al escribirlo e inutilizar tu sistema, por lo que antes de ejecutar el comando **strip** posiblemente sea buena idea hacer una copia de respaldo del sistema LFS en su estado actual.

Antes de hacer la eliminacón de símbolos, se ha de tener mucho cuidado para asegurar que no se esté ejecutando ningún binario que vaya a ser procesado. Si no estás seguro de si entraste al entorno chroot con el comando mostrado en la Sección 6.4, “Entrar al entorno chroot”, entonces sal primero del chroot:

```
logout
```

Luego vuelve a entrar con:

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

Ahora puedes procesar con tranquilidad los binarios y librerías:

```
/tools/bin/find /{,usr/}{bin,lib,sbin} -type f \
  -exec /tools/bin/strip --strip-debug '{}' ';'
```

Se avisará de que no se reconoce el formato de un buen número de ficheros. Puedes ignorar esos avisos, sólo indican que se trata de guiones en vez de binarios.

Si el espacio en disco es escaso, se puede usar la opción *--strip-all* sobre los binarios que hay en /{,usr/}{bin,sbin} para ganar varios megabytes más. Pero no uses dicha opción sobre las librerías: las destruirías.

6.60. Limpieza

A partir de ahora, cuando salgas del entorno chroot y desees entrar de nuevo en él, deberás ejecutar el siguiente comando chroot modificado:

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

La razón para esto es que ya no son necesarios los programas que hay en /tools. Puesto que ya no son necesarios, puedes borrar el directorio /tools si lo deseas.

Nota

Al eliminar /tools también se eliminan las copias temporales de Tcl, Expect y DejaGnu que fueron usadas para ejecutar los bancos de pruebas. Si quieres usar estos programas más adelante, necesitarás recompilarlos y reinstalarlos. El libro BLFS tiene instrucciones para esto (mira *http://www.linuxfromscratch.org/blfs/*).

Si los sistemas de ficheros virtuales han sido desmontados, ya sea manualmente o debido a un reinicio, asegurate de que se encuentran montados cuando reentres al chroot. Este proceso se esplicó en Sección 6.2.2, “Montar y poblar /dev” y Sección 6.2.3, “Montar los sistemas de ficheros virtuales del núcleo”.

Capítulo 7. Configurar los guiones de arranque del sistema

7.1. Introducción

Este capítulo detalla cómo instalar y configurar el paquete LFS-Bootscripts. Muchos de estos guiones funcionarán sin necesidad de modificarlos, pero algunos necesitan ficheros de configuración adicionales, pues manejan información dependiente del hardware.

En este libro se utilizan guiones de inicio al estilo System-V porque son ampliamente utilizados. Para consultar otras opciones, una receta que detalla la configuración del inicio al estilo BSD se encuentra en *http://www.linuxfromscratch.org/hints/downloads/files/bsd-init.txt*. Buscando “depinit” en las listas de correo de LFS encontrarás otra alternativa.

Si decides usar algún otro estilo de guiones de inicio, sáltate este capítulo y pasa al Capítulo 8.

7.2. LFS-Bootscripts-20070813

El paquete LFS-Bootscripts contiene un conjunto de guiones para iniciar/parar el sistema LFS durante el arranque/apagado.

Tiempo estimado de construcción: less than 0.1 SBU

Espacio requerido en disco: 0.4 MB

7.2.1. Instalación de LFS-Bootscripts

Instala el paquete:

```
make install
```

7.2.2. Contenido de LFS-Bootscripts

Guiones instalados: checkfs, cleanfs, console, consolelog, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountkernfs, network, rc, reboot, sendsignals, setclock, static, swap, sysctl, sysklogd, template, udev, and udev_retry

Descripciones cortas

checkfs Comprueba la integridad de los sistemas de ficheros antes de ser montados (con la excepción de los que usan registros de transacciones [journal] o los que se montan desde la red).

cleanfs Elimina los ficheros que no deben guardarse cuando se arranca de nuevo el sistema, como aquellos en `/var/run/` y `/var/lock/`. Regenera `/var/run/utmp` y elimina los ficheros `/etc/nologin`, `/fastboot` y `/forcefsck` si existen.

console Carga el mapa de teclado correcto para el tipo de teclado deseado. También establece la fuente de pantalla.

consolelog Establece el nivel de registro del núcleo para los mensajes de control mostrados en la consola.

functions Contiene funciones comúnes, como la comprobación de errores y estado, usadas por diferentes guiones de arranque.

halt Cierra el sistema.

ifdown Ayuda al guión network para detener los dispositivos de red.

ifup Ayuda al guión network para iniciar los dispositivos de red.

localnet Establece el nombre de máquina usado por el sistema (hostname) y activa el dispositivo de red interna (loopback).

modules Carga los módulos del núcleo listados en `/etc/sysconfig/modules`, usando los argumentos que se indiquen allí.

mountfs Monta todos los sistemas de ficheros que no estén marcados como *noauto* o que no se monten a través de la red.

mountkernfs Monta los sistemas de ficheros virtuales del núcleo, como `/proc`.

network Activa las interfaces de red, como las tarjetas de red, y establece la puerta de enlace por defecto (gateway) cuando es necesario.

rc El controlador maestro de los niveles de arranque. Es el responsable de lanzar todos los demás guiones de arranque, uno a uno, en una secuencia determinada por el nombre del enlace simbólico a procesar.

reboot Reinicia el sistema.

sendsignals Se asegura de que todos los procesos terminen antes de parar o reiniciar el sistema.

setclock Fija el reloj del núcleo a la hora local en caso de que el reloj del ordenador no esté fijado a la hora UTC.

static Suministra la funcionalidad necesaria para asignar una dirección IP estática a una interfaz de red.

swap Activa y desactiva las particiones y ficheros de intercambio (swap).

sysctl Carga los valores de configuración del sistema desde `/etc/sysctl.conf`, si este fichero existe, dentro del núcleo en ejecución

sysklogd Lanza y detiene los demonios de registro de eventos del sistema y del núcleo.

template Una plantilla para crear guiones de arranque personalizados para otros demonios.

udev Prepara el directorio `/dev` e inicia Udev.

udev_retry Reintenta eventos udev fallidos y copia los ficheros de reglas generados de `/dev/.udev` a `/etc/udev/rules.d` si se solicita.

7.3. ¿Cómo funcionan los guiones de arranque?

Linux utiliza como sistema de inicio SysVinit, que se basa en el concepto de *niveles de ejecución*. Este sistema de inicio puede variar ampliamente de un sistema a otro, por lo tanto no se debe asumir que porque las cosas funcionen en una distribución en concreeto tengan que funcionar en LFS también. LFS tiene su propia manera de hacer las cosas, la cual suele respetar los estándares aceptados.

SysVinit (al que llamaremos "init" a partir de este momento) se basa en un esquema de niveles de ejecución. Hay 7 (numerados del 0 al 6) niveles de ejecución (en realidad existen más, pero son para casos especiales y es raro utilizarlos. Mira `init(8)` para más detalles) y cada uno de ellos indica lo que debe hacer el sistema durante el arranque. El nivel de ejecución por omisión es el 3. He aquí una breve descripción de los distintos niveles de ejecución como suelen implementarse:

```
0: parada del sistema
1: modo monousuario
2: modo multiusuario sin red
3: modo multiusuario con red
4: reservado para personalizar, si no, hace lo mismo que el 3
5: Igual que el 4. Normalmente se utiliza para iniciar el entorno
   gráfico (mediante xdm de X o kdm de KDE)
6: reinicio del sistema
```

Para cambiar el nivel de ejecución se utiliza el comando **init *<nivel de ejecución>*** donde *<nivel de ejecución>* representa el nivel de ejecución que se desea arrancar. Por ejemplo, para reiniciar el sistema se utilizaría el comando **init 6**. El comando **reboot** no es más que un alias de dicho comando, al igual que el comando **halt** lo es de **init 0**.

Debajo de `/etc/rc.d` existe una serie de directorios `rc?.d` (donde ? representa el número del nivel de ejecución), más el directorio `rcsysinit.d`, que contienen un conjunto de enlaces simbólicos. Los nombres de estos enlaces simbólicos empiezan con K o con S seguidos de 2 cifras. Los enlaces que comienzan por una *K* indican la parada (kill) de un servicio, mientras que la *S* indica su inicio (start). Las dos cifras determinan el orden de ejecución, desde 00 hasta 99; cuanto menor sea el número, antes se ejecutará. Cuando **init** cambia a otro nivel de ejecución, los servicios apropiados son iniciados o parados, dependiendo del nivel de ejecución elegido.

Los guiones reales se enuentran en `/etc/rc.d/init.d`. Ellos son los que hacen el trabajo y todos los enlaces simbólicos apuntan a ellos. Los enlaces de parada e inicio apuntan al mismo guión en `/etc/rc.d/init.d`. Esto se debe a que los guiones pueden invocarse con parámetros diferentes como *`start`*, *`stop`*, *`restart`*, *`reload`* y *`status`*. Cuando se encuentra un enlace K, se ejecuta el guión apropiado con el argumento *`stop`*. Cuando se encuentra un enlace S, se ejecuta el guión apropiado con el argumento *`start`*.

Hay una excepción a esta explicación. Los enlaces que comienzan por *S* en los directorios `rc0.d` y `rc6.d` no inician nada. Estos guiones se invocan siempre con el parámetro *`stop`* para parar algo. La lógica tras esto es que cuando el usuario va a parar o reiniciar el sistema no es necesario iniciar nada. El sistema sólo necesita ser detenido.

He aquí una descripción de lo que hace cada parámetro:

`start`
: Inicia el servicio.

`stop`
: Para el servicio.

`restart`
: El servicio se para y se vuelve a iniciar.

reload
: Se actualiza la configuración del servicio. Este parámetro se utiliza tras la modificación del fichero de configuración cuando no es necesario reiniciar el servicio.

status
: Dice si el servicio se está ejecutando y con qué identificador de proceso (PID).

Eres libre de modificar la forma en que funciona el proceso de arranque (después de todo es tu propio sistema LFS). Los ficheros aquí mostrados son un ejemplo de cómo puede hacerse.

7.4. Manejo de dispositivos y módulos en un sistema LFS

En el Capítulo 6 se instaló el paquete Udev. Antes de entrar en detalles sobre cómo funciona, repasaremos los anteriores métodos de manejo de dispositivos.

Tradicionalmente, los sistemas Linux en general utilizan un método estático de creación de dispositivos, implicando que un gran número de nodos de dispositivo son creados en `/dev` (literalmente, cientos de nodos) sin tener en cuenta si el dispositivo hardware correspondiente existe en realidad. Esto se hace típicamente mediante un guión **MAKEDEV**, que contiene una serie de llamadas al programa **mknod** con los números mayor y menor correspondientes a cada posible dispositivo que pudiera existir en el mundo.

Con el uso del método Udev, sólo se crearán los nodos correspondientes a aquellos dispositivos detectados por el núcleo. Debido a que estos nodos de dispositivo se crearán cada vez que se inicie el sistema, se almacenarán en un sistema de ficheros `tmpfs` (el cual existe por completo en memoria). Los nodos de dispositivo no necesitan mucho espacio, por lo que la memoria utilizada es muy poca.

7.4.1. Historia

En Febrero de 2000, un nuevo sistema de ficheros llamado `devfs` fue incluido en los núcleos 2.3.46 y estuvo disponible en la serie 2.4 de los núcleos estables. Aunque estaba presente en las propias fuentes del núcleo, este método de creación dinámica de dispositivos nunca recibió mucho apoyo por parte del equipo de desarrolladores del núcleo.

El principal problema con el sistema adoptado por `devfs` era el modo en el que manejaba la detección, creación y denominación de dispositivos. El último punto, la denominación de los nodos, fue quizás el más crítico. Está generalmente aceptado que si los nombres de dispositivos son configurables, entonces las políticas de denominación deberían ser establecidas por un administrador del sistema y no impuestas por un desarrollador en particular. El sistema de ficheros `devfs` sufre también de extraños comportamientos inherentes a su diseño y que no pueden corregirse sin una revisión sustancial del núcleo. Durante un tiempo fué marcado como descartado debido a la falta de mantenimiento, siendo removido finalmente del núcleo en Junio de 2006.

Con el desarrollo del árbol inestable 2.5 del núcleo, posteriormente liberado como núcleos estables de la serie 2.6, aparece un nuevo sistema de ficheros virtual llamado `sysfs`. El trabajo de `sysfs` es exportar una visión de la configuración hardware del sistema a los procesos de usuario. Con esta representación visible a nivel de usuario, la posibilidad de encontrar un sustituto para `devfs` a nivel de usuario se hace mucho más real.

7.4.2. Implementación de Udev

7.4.2.1. Sysfs

Arriba se mencionó brevemente el sistema de ficheros `sysfs`. Uno podría preguntarse cómo conoce `sysfs` los dispositivos presentes en el sistema y qué números de dispositivo debe usar. Los crontroladores que se han compilado directamente dentro del núcleo registran sus objetos en `sysfs` a medida que son detectados por el núcleo. Para los

controladores compilados como módulos, esto sucederá cuando se cargue el módulo. Una vez montado el sistema de ficheros `sysfs` (en `/sys`), los datos registrados en `sysfs` por los controladores están disponibles para los procesos de usuario y para que **udevd** cree los nodos de dispositivo.

7.4.2.2. Guión de inicio de Udev

El guión de inicio **S10udev** se ocupa de la creación de dichos nodos de dispositivo cuando se inicia Linux. Este guión desactiva **/sbin/hotplug** como manejador uevent. Esto se hace debido a que el núcleo ya no necesita llamar a binarios externos. En su lugar, **udevd** escuchará en un conector de red los uevent que el núcleo genere. A continuación, el guión de arranque copia los nodos de dispositivo estáticos que encuentre en `/lib/udev/devices` a `/dev`. Esto es necesario ya que algunos dispositivos, directorios y enlaces simbólicos son requeridos antes de que el proceso de manejo dinámico de dispositivos esté disponible durante las primeras fases del arranque del sistema. La creación de nodos de dispositivo estáticos en `/lib/udev/devices` proporciona también una solución para aquellos dispositivos no soportados por la infraestructura de manejo dinámico de dispositivos. Entonces el guión de arranque iniciará el demonio Udev, **udevd**, que actuará sobre cualquier uevent recibido. Por último, el guión de arranque fuerza al núcleo a repetir los uevents de los dispositivos que ya hayan sido registrados y espera a que **udevd** los maneje.

7.4.2.3. Creación de nodos de dispositivo

Para obtener los números mayor y menor correctos de un dispositivo, Udev utiliza la información proporcionada por `sysfs` en `/sys`. Por ejemplo, `/sys/class/tty/vcs/dev` contiene la cadena “7:0”. Esta cadena es usada por **udevd** para crear un nodo de dispositivo con número mayor *7* y menor *0*. Los permisos y modos de los nodos creados en el directorio `/dev` son determinados por las reglas especificadas en los ficheros que hay en el directorio `/etc/udev/rules.d/`. Estas se encuentran numeradas en un formato similar al del paquete LFS-Bootscripts. Si **udevd** no puede encontrar una regla para el dispositivo que está creando, utilizará los permisos *660* y como propietario *root:root*.La documentación sobre la sintaxis de los ficheros de reglas de configuración de Udev se encuentra en `/usr/share/doc/udev-113/index.html`

7.4.2.4. Manejo de módulos

Los controladores de dispositivos compilados como módulos pueden tener alias dentro de ellos. Los alias son visibles en la salida del programa **modinfo** y normalmente están relacionados con el identificados específico del bus de los dispositivos soportados por el módulo. Por ejemplo, el controlador *snd-fm801* soporta dispositivos PCI con ID de vendedor 0x1319 e ID de dispositivo 0x0801, y tiene el alias “pci:v00001319d00000801sv*sd*bc04sc01i*”. Para muchos dispositivos, el controlador del bus exporta a través de `sysfs` el alias del controlador que podría manejar el dispositivo. Por ejemplo, el fichero `/sys/bus/pci/devices/0000:00:0d.0/modalias` podría contener la cadena “pci:v00001319d00000801sv00001319sd00001319bc04sc01i00”. Las reglas instaladas por LFS harán que **udevd** llame a **/sbin/modprobe** con el contenido de la variable de entorno uevent `MODALIAS` (que debería ser igual al contenido del fichero `modalias` del sysfs), cargando todos los módulos cuyos alias concuerden con dicha cadena tras la expansión de comodines.

En el ejemplo, esto significa que en adición a *snd-fm801*, el obsoleto (y no deseado) controlador *forte* será cargado si está disponible. Más abajo se muestran formas de evitar la carga de dispositivos no deseados.

El propio núcleo también es capaz de cargar módulos para protocolos de red, sistemas de ficheros y soporte NLS bajo demanda.

7.4.2.5. Manejo de dispositivos dinámicos/conectables en caliente

Cuando conectas un dispositivo, como un reproductor MP3 por USB, el núcleo reconoce que el dispositivo está conectado ahora y genera un uevent. Este uevent es manejado por **udevd** como se describe arriba.

7.4.3. Problemas con la carga de módulos y la creación de dispositivos

Hay algunos problemas en relación con la creación automática de nodos de dispositivos.

7.4.3.1. Un módulo del núcleo no se carga automáticamente

Udev cargará un módulo sólo si este tiene un alias específico del bus y el controlador del bus exporta correctamente los alias necesarios a `sysfs`. En caso contrario, la carga del módulo deberá realizarse por otros métodos. Con Linux-2.6.22.6, se sabe que Udev carga los controladores correctamente escritos para dispositivos INPUT, IDE, PCI, USB, SCSI, SERIO y FireWire.

Para determinar si el controlador de dispositivo que necesitas tiene el soporte necesario para Udev, ejecuta **modinfo** con el nombre del módulo como argumento. Ahora intenta localizar el directorio del dispositivo bajo `/sys/bus` y comprueba si hay un fichero `modalias`.

Si el fichero `modalias` existe en `sysfs`, el controlador soporta el dispositivo y puede hablar con él directamente, pero no contiene el alias, esto es un fallo en el controlador. Carga el controlador sin la ayuda de Udev y espera que el problema sea solucionado más adelante.

Si no hay un fichero `modalias` en el directorio correspondiente bajo `/sys/bus`, esto significa que los desarrolladores del núcleo no han añadido todavía soporte de alias a ese tipo de bus. Con Linux-2.6.22.6, este es el caso con los bus ISA. Se espera que esto se resuelva e futuras versiones del núcleo.

Udev no está pensado para cargar controladores “envoltorio”, como *snd-pcm-oss* o controladores que no pertenecen al hardware, como *loop*.

7.4.3.2. Un módulo del núcleo no se carga automáticamente y Udev no intenta cargarlo

Si el módulo “envoltorio” sólo amplía la funcionalidad proporcionada por otro módulo (del modo que *snd-pcm-oss* amplía la funcionalidad de *snd-pcm* haciendo que las tarjetas de sonido están disponibles para aplicaciones OSS), configura **modprobe** para cargar el envoltorio despues de que Udev carge el módulo envuelto. Para hacer esto, añade una línea “install” en `/etc/modprobe.conf`. Por ejemplo:

```
install snd-pcm /sbin/modprobe -i snd-pcm ; \
    /sbin/modprobe snd-pcm-oss ; true
```

Si el módulo en cuentión no es un envoltorio y es útil por si mismo. configura el guión de arranque **S05modules** para cargar dicho módulo en el arranque del sistema. Para hacer esto, añade una nueva línea con el nombre del módulo al fichero `/etc/sysconfig/modules`. Esto también funciona para módulos envoltorio, pero no es óptimo en dicho caso.

7.4.3.3. Udev carga módulos no deseados

Entonces no construyas el módulo, o ignoralo en el fichero `/etc/modprobe.conf` como se hace en el siguiente ejemplo para el módulo *forte*:

```
blacklist forte
```

Los módulos ignorados aún pueden cargarse manualmente usando el comando **modprobe**.

7.4.3.4. Udev crea un dispositivo incorrectamente, o hace un enlace simbólico erroneo

Esto sucede normalmento cuando una regla concuerda inesperadamente con un dispositivo. Poe ejemplo, una regla pobremente escrita puede coincidir, en cuanto al vendedor, tanto con un disco SCSI (como se desea) como con el correspondiente dispositivo SCSI genérico (incorrecto). Encuentra la regla erronea y hazla más específica.

7.4.3.5. Udev no crea un dispositivo

Se asume que el controlador ha sido compilado estáticamente dentro del núcleo o que ya ha sido cargado como módulo, y que ya has comprobado que Udev no crea un dispositivo equivocado.

Udev no tiene la información necesaria para crear un nodo de dispositivo si un controlador del núcleo no exporata sus datos a `sysfs`. Este es el caso más común con controladores externos al árbol del núcleo. Crea un nodo de dispositivo estático en `/lib/udev/devices` con los números mayor/menor apropiados (mira el fichero `devices.txt` en la documentación del núcleo o la documentación proporcionada por el distribuidor del controlador externo). El nodo de dispositivo estático será copiado a `/dev` por el guión de arranque **S10udev**.

7.4.3.6. El orden de denominación de los dispositivos cambia aleatoriamente tras un reinicio

Esto se debe al hecho de que Udev, por diseño, maneja los uevent y la carga de módulos en paraleo, y por tanto en un orden impredecible. Esto nunca será "fijado". No deberías confiar en que los nombres de dispositivos del núcleo sean estables. En su lugar, crea tus propias reglas para crear enlaces simbólicos con nombres estables basadas en algún atributo estable del dispositivo, como un número de serie o la salida de las diversas utilidades *_id instaladas por Udev. Mira los ejemplos en Sección 7.12, "Crear enlaces simbólicos persnalizados a los dispositivos" y Sección 7.13, "Configuración del guión network".

7.4.4. Lectura útil

En los siguientes sitios hay documentación de ayuda adicional:

- Una implementación de `devfs` en espacio de usuario *http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf*
- FAQ de Udev *http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ*
- El sistema de ficheros `sysfs` *http://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf*

7.5. Configuración del guión setclock

El guión **setclock** lee la hora del reloj interno del ordenador, conocido también como reloj BIOS o CMOS (Semiconductor de Oxido de Metal Complementario). Si el reloj hardware está establecido a la hora UTC, este guión la convierte a la hora local mediante el fichero `/etc/localtime` (que le indica al programa **hwclock** en qué zona horaria se encuentra el usuario). No hay manera de detectar automáticamente si el reloj utiliza UTC o no, así que esto se debe configurar manualmente.

Si no puedes recordar si el reloj hardware está en UTC o no, averígualo ejecutando el comando **`hwclock --localtime --show`**. Esto mostrará la hora actual según el reloj hardware. Si dicha hora coincide con la de tu reloj, entonces el reloj hardware está a la hora local. Si la salida de **hwclock** no es la hora local, seguramente esté en la hora UTC. Verifica esto añadiendo o restando la cantidad de horas correspondiente a tu zona local a la hora mostrada por **hwclock**. Por ejemplo, si vives en la zona horaria MST, conocida también como GMT -0700, añade siete horas a la hora local.

Cambia abajo el valor de la variable `UTC` a *`0`* (cero) si el reloj hardware *no* utiliza la hora UTC.

Crea un nuevo fichero `/etc/sysconfig/clock` ejecutando lo siguiente:

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

UTC=1

# End /etc/sysconfig/clock
EOF
```

En *http://www.linuxfromscratch.org/hints/downloads/files/time.txt* hay disponible una buena receta que trata sobre la hora en LFS. En ella se explican conceptos como las zonas horarias, UTC y la variable de entorno `TZ`.

7.6. Configurar la consola Linux

Esta sección explica cómo configurar los guiones de arranque **console** y **consolelog**, que establecen el mapa del teclado, la fuente de consola y el nivel de registro de la consola del núcleo. Si no se van a utilizar caracteres no ASCII (como por ejemplo la Libra inglesa y el Euro) y el teclado es del tipo U.S., puedes saltarte gran parte de esta sección. Sin el fichero de configuración el guión de inicio **console** no hará nada.

Los guiones **console** y **consolelog** utilizan `/etc/sysconfig/console` como fichero de configuración. Decide qué mapa de teclado y fuente de pantalla se usarán. Los diversos CÓMO específicos para cada idioma pueden ayudarte en esto, consulta *http://www.tldp.org/HOWTO/HOWTO-INDEX/other-lang.html*. Si aún tienes dudas, mira en el directorio `/lib/kbd` los mapas de teclados y fuentes de pantalla válidos. Lee las páginas de manual `loadkeys(1)` y `setfont(8)` para determinar los argumentos correctos para estos programas.

El fichero `/etc/sysconfig/console` debería contener líneas del tipo: VARIABLE="valor". Se reconocen las siguientes variables:

LOGLEVEL
: Esta variable especifica el nivel de registro para los mensajes que el núcleo envía a la consola según establece **dmesg**. Los niveles válidos van desde "1" (no mensajes) hasta "8". El nivel por defecto es "7".

KEYMAP
: Esta variable especifica los argumentos para el programa **loadkeys**, típicamente el nombre del teclado a cargar, por ejemplo “es”. Si no se establece esta variable, el guión de arranque no ejecutará el programa **loadkeys**, y se usará el mapa de teclado por defecto.

KEYMAP_CORRECTIONS
: Esta variable (usada en raros casos) especifica los argumentos para la segunda llamada al programa **loadkeys**. Es útil si el mapa del teclado no es completamente satisfactorio y deben hacerse pequeños ajustes. Por ejemplo, para incluir el símbolo del Euro en un mapa de teclado que no lo tiene, establece esta variable a “euro2”.

FONT
: Esta variable especifica los argumentos para el programa **setfont**. Típicamente, esto incluye el nombre de la fuente, “-m” y el nombre de la aplicación de mapa de caractéres a cargar. Por ejemplo, para cargar la fuente “lat1-16” junto con la aplicación de mapa de caractéres “8859-1” (que es lo correcto en USA), establece esta variable a “lat1-16 -m 8859-1”. Si no se establece la variable, el guión de arranque no lanzará el programa **setfont** y se usará la fuente VGA por defecto junto con la aplicación de mapa de caractéres por defecto.

UNICODE
: Establece esta variable a “1”, “yes” o “true”para poner la consola en modo UTF-8. Es útil en locales basadas en UTF-8 e inofensiva en el resto.

LEGACY_CHARSET

Para muchos esquemas de teclado no hay un mapa de teclado Unicode base en el paquete Kdb. El guión de arranque **console** convertirá al vuelo un mapa de teclado disponible a UTF-8 si esta variable se establece a la codificación del mapa de teclado no UTF-8 disponible.

Algunos ejemplos:

- Para una configuración no Unicode, normalmente sólo son necesarias las variables KEYMAP y FONT. Por ejemplo, para configurar el polaco podría usarse:

```
cat > /etc/sysconfig/console << "EOF"
# Inicio de /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# Fin de /etc/sysconfig/console
EOF
```

- Como se mencionó anteriormente, a veces es necesario ajustar ligeramente el mapa de taclado base. El siguiente ejemplo añade el símbolo del Euro al mapa de teclado alemán:

```
cat > /etc/sysconfig/console << "EOF"
# Inicio de /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"

# Fin de /etc/sysconfig/console
EOF
```

- Lo siguiente es un ejemplo de búlgaro con Unicode activado, donde el mapa de teclado base UTF-8 existe:

```
cat > /etc/sysconfig/console << "EOF"
# Inicio de /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# Fin de /etc/sysconfig/console
EOF
```

- Debido al uso de una fuente LatArCyrHeb-16 de 512-glifos en el ejempo anterior, los colores brillantes no estarán disponibles en la consola Linux a menos que se utilice framebuffer. Si uno quiere tener colores brillantes sin framebuffer y puede vivir sin los caractéres que no pertenecen a su idioma, es posible usar una fuente de 256-glifos especifica para el idioma, como se muestra a continuación:

```
cat > /etc/sysconfig/console << "EOF"
# Inicio de /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# Fin de /etc/sysconfig/console
EOF
```

- No existe un mapa de teclado UTF-8 preparado para ruso, por tanto este debe generarse mediante la conversión del mapa de teclado KOI8-R, como se muestra a continuación:

```
cat > /etc/sysconfig/console << "EOF"
# Inicio de /etc/sysconfig/console

UNICODE="1"
KEYMAP="ru_ms"
LEGACY_CHARSET="koi8-r"
FONT="LatArCyrHeb-16"

# Fin de /etc/sysconfig/console
EOF
```

- Algunos mapas de teclado tienen teclas muertas (es decir, teclas que no producen un caracter por si mismas, pero añaden un acento a la tecla pulsada a continuación) o definen reglas de composición (como “pulsa Ctrl+. A E para obtener Æ” en el mapa de teclado por defecto). En modo de teclado UTF-8 Linux-2.6.22.6 asume que los caracteres acentuados mediante teclas muertas o composición se encruentran en el rango Latin-1 de Unicode, y es imposible cambiar dicha asuncion. Por tanto, los caracteres acentuados necersarios para, por ejemplo, el checo, no pueden teclearse en una consola Linux en modo UTF-8 (pero los ficheros que contienen dichos caractéres pueden mostrarse correctamente). Por tanto la solución es o evitar el uso de UTF-8 o instalar el sistema de ventanas X, que no tienes estas limitaciones en su manejo de entrada.
- Para chino, japonés, coreano y algunos idiomas mas, la consola Linux no puede configurarse para mostrar los caractéres necesarios. Los usuarios que necesiten dichos idiomas deberían instalar el sistema X Window, fuentes que cubran los rangos de caractéres necesarios, y el método de entrada adecuado (por ejemplo, SCIM soporta una gran variedad de idiomas).

Nota

El fichero `/etc/sysconfig/console` sólo controla la consola de texto Linux. No tiene nada que ver con establecer el mapa de teclado y fuentes de terminal correctas en el sistema X Window, de sesiones ssh o de una consola serie. En dichas situaciones, las limitaciones mencionadas en los últimos dos puntos anteriores no son aplicables.

7.7. Configuración del guión sysklogd

El guión `sysklogd` invoca al programa **syslogd** con la opción *-m 0*. Esta opción deshabilita la marca de tiempo periódica que **syslogd** escribe por defecto en el fichero de registro cada 20 minutos. Para habilitar esta marca de tiempo periódica, edita el guión `sysklogd` y realiza los cambios necesarios. Para más información mira **`man syslogd`**.

7.8. Crear el fichero /etc/inputrc

El fichero `/etc/inputrc` se ocupa del mapeado del teclado para situaciones concretas. Este fichero es el fichero de inicio usado por Readline, la librería para cuestiones de entrada usada por Bash y otros intérpretes de comandos.

Generalmente los usuarios no necesitan mapeados específicos del teclado, por lo que el siguiente comando crea un `/etc/inputrc` global usado por todo el que ingrese en el sistema. Si más tarde decides que necesitas modificarlo para cada usuario, puedes crear un fichero `.inputrc` en el directorio del usuario con el mapeado modificado.

Para más información sobre cómo editar el fichero `inputrc`, mira **info bash**, sección *Readline Init File* (Fichero de Inicio de Readline). **info readline** es también una buena fuente de información.

A continuación hay un `/etc/inputrc` global genérico, con comentarios para explicar lo que hace cada opción. Advierte que los comentarios no pueden estar en la misma línea que los comandos. Crea el fichero usando el siguiente comando:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

7.9. Los ficheros de inicio de Bash

El intérprete de comandos **/bin/bash** (al que nos referiremos como “el intérprete”) utiliza una colección de ficheros de inicio para ayudar a crear un entorno de trabajo. Cada fichero tiene un uso específico y pueden generar diferentes entornos de ingreso o interactivos. Los ficheros del directorio `/etc` proporcionan ajustes globales. Si existe un fichero diferente en el directorio personal, este puede sobreescribir los ajustes globales.

Un intérprete de ingreso interactivo se inicia tras ingresar en el sistema, usando **/bin/login**, mediante la lectura del fichero `/etc/passwd`. Un intérprete interactivo de no ingreso se inicia en la línea de comandos (es decir, `[prompt]$` **/bin/bash**). Un intérprete no interactivo está presente usualmente cuando se ejecuta un guión del intérprete de comandos. Es no interactivo porque está procesando un guión y no esperando indicaciones del usuario entre comandos.

Para más información, consulta en **info bash** la sección *Bash Startup Files and Interactive Shells* (Ficheros de inicio de Bash e intérpretes interactivos).

Los ficheros `/etc/profile` y `~/.bash_profile` son leidos cuando el intérprete se invoca como un intérprete interactivo de ingreso.

El siguiente fichero `/etc/profile` básico establece algunas variables de entorno necesarias para el soporte de idioma nativo. Al establecerlas correctamente se obtiene:

- La salida de los programas traducida al idioma nativo.
- Correcta clasificación de los caracteres en letras, dígitos y otros tipos. Esto es necesario para que **bash** acepte correctamente los caracteres no ASCII en la línea de comandos en idiomas diferentes al inglés.
- La correcta ordenación alfabética propia del país.
- Un apropiado tamaño de papel por defecto.
- Un formato correcto para los valores monetarios, horarios y fechas.

Sustituye a continuación *<ll>* con el código de dos letras del idioma deseado (por ejemplo, “en”) y *<CC>* con el código de dos letras de tu país (por ejemplo, “GB”). *<charmap>* debe reemplazarse por el nombre canónico del mapa de caracteres para tu locale elegida. También pueden estar presentes modificadores opcionales como “@euro”.

Puedes obtener la lista de todas las locales soportadas por Glibc ejecutando el siguiente comando:

```
locale -a
```

Los mapas de caractéres pueden tener una serie de sinónimos, por ejemplo “ISO-8859-1” se referencia también como “iso8859-1” y “iso88591”. Algunas aplicaciones no pueden manejar correctamente los diversos sinónimos (poe ejemplo, necesitan que “UTF-8” se escriba como “UTF-8”, no “utf8”), por lo que es más seguro elegir el nombre canónico de la locale. Para determinar el nombre canónico, en el que *<nombre de la locale>* es la salida mostrada por **locale -a** para tu locale preferida (“en_GB.iso88591” en nuestro ejemplo).

```
LC_ALL=<nombre de la locale> locale charmap
```

Para la locale “en_GB.iso88591”, el anterior comando mostrará:

```
ISO-8859-1
```

Esto resulta en un ajuste final para la locale de “en_GB.ISO-8859-1”. Es importante que la locale encontrada usando el método anterior sea comprobada antes de añadirla a los ficheros de inicio de Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

Dichos comandos deberán mostrar los nombres del idioma, la codificación de caracteres usada por la locale, el símbolo de la moneda local y el prefijo a marcar antes del número de teléfono para acceder al país. Si cualquiera de los comandos anteriones fallase con un mensaje similar al mostrado a continuación, esto significa que o tu locale no se instaló en el Capítulo 6, o que no está soportada por la instalación por defecto de Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Si esto sucede, deberías o bien instalar la locale deseada usando el comando **localedef**, o considerar la elección de una locale diferente. El resto de instrucciones asumen que no hay mensajes de error procedentes de Glibc.

Algunos paquetes más allá del LFS puede que no tengan soporte para tu locale elegida. Un ejemplo es la librería X (que es parte del sistem X Window), que mostrará el siguiente mensaje de error si la locale no coincide exactamente con uno de los nombres de mapa de caractéres de sus ficheros internos:

```
Warning: locale not supported by Xlib, locale set to C
```

En ciertos casos Xlib espera que el mapa de caracteres sea listado en mayúsculas y con guiones. Por ejemplo, "ISO-8859-1" en vez de "iso88591". También es posible encontrar una especificación apropiada eliminando de la especificación de la locale la parte del mapa de caracteres. Esto puede cmprobarse ejecutando el comando **locale charmap** en ambas locales. Por ejemplo, podrías tener que cambiar "de_DE.ISO-8859-15@euro" por "de_DE@euro" para hacer que Xlib reconociese esta locale.

Otros paquetes también pueden funcionar incorrectamente (pero no necesariamente mostrar un mensaje de error) si el nombre de la locale no cumple sus especificaciones. En estos casos, investigar cómo otras distribuciones Linux soportan tu locale podría proporcionar información útil.

Una vez hayas determinado los ajustes correctos para el idioma, crea el fichero `/etc/profile`:

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

export LANG=<ll>_<CC>.<charmap><@modifiers>

# End /etc/profile
EOF
```

Las locales “C” (la que se tiene por defecto) y “en_US” (la recomendada para los usuarios de habla inglesa de los Estados Unidos) son diferentes. “C” utiliza el grupo de caractéres US-ASCII de 7-bits, y trata bytes con el bit alto establecido como caracter inválido. Esto es por lo que, por ejemplo, el comando **ls** los sustituye con interrogantes en dicha locale. También, un intento de enviar correo con dichos caractéres desde Mutt o Pine resulta en que se envie un mensaje de no conformancia con RFC (el grupo de caractéres en el mensaje de salida se indoca como “desconocido de 8-bit”). Por tanto puedes usar la locale “C” sólo si estás seguro de que núnca necesitarás caractéres de 8-bits.

Las locales basadas en UTF-8 no están bien soportadas por muchos programas. Por ejemplo, el programa **watch** mustra sólo caractéres ASCII en locales UTF-8 y no tiene dicha restricción en locales tradicionales de 8-bits como en_US. Se está trabajando en documentar, y si es posible corregir, dichos problemas. Mira *http://www.linuxfromscratch.org/blfs/view/svn/introduction/locale-issues.html*.

7.10. Configuración del guión localnet

Parte del trabajo del guión **localnet** es establecer el nombre de la máquina. Esto se configura en el fichero `/etc/sysconfig/network`.

Crea el fichero `/etc/sysconfig/network` e introduce el nombre de tu máquina ejecutando:

```
echo "HOSTNAME=<lfs>" > /etc/sysconfig/network
```

Debes substituir *<lfs>* por el nombre con el que debe de conocerse tu máquina. No escribas el FQDN (nombre completo de la máquina, incluido su dominio). Esa información la escribiremos más tarde en el fichero `/etc/hosts`

7.11. Personalizar el fichero /etc/hosts

Si se va a configurar una tarjeta de red, decide la dirección IP, el nombre de dominio cualificado (FQDN) y los posibles alias para escribirlos en el fichero `/etc/hosts`. La sintaxis es:

```
dirección_IP miordenador.example.org alias
```

A no ser que tu computadora sea visible en Internet (es decir, tengas un dominio registrado y asignado un bloque de direcciones IP válido, la mayoría no tenemos esto), deberías asegurarte de que la dirección IP queda dentro del rango de direcciones IP de la red privada. Los rangos válidos son:

```
Rango de direcciones de la red privada  Prefijo Normal
10.0.0.1 - 10.255.255.254                8
172.x.0.1 - 172.x.255.254               16
192.168.y.1 - 192.168.y.254             24
```

x pude ser cualquier número en el rango 16-31. y puede ser cualquier número en el rango 0-255.

Una dirección IP privada válida podría ser 192.168.1.1. Un FQDN válido para esta IP podría ser lfs.example.org.

Aunque no uses una tarjeta de red, un FQDN válido es requerido. Este es necesario para que ciertos programas funcionen correctamente.

Crea el fichero `/etc/hosts` ejecutando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
<192.168.1.1> <HOSTNAME.example.org> [alias1] [alias2 ...]

# End /etc/hosts (network card version)
EOF
```

Debes cambiar los valores *<192.168.1.1>* y *<HOSTNAME.example.org>* por los tuyos específicos o los requeridos (si la máquina estará conectada a una red ya existente y el administrador de la red/sistema es el que asigna una dirección IP). Los alias son opcionales y pueden omitirse.

Si no se va a configurar una tarjeta de red, crea el fichero `/etc/hosts` ejecutando:

```
cat > /etc/hosts << "EOF"
# Begin /etc/hosts (no network card version)

127.0.0.1 <HOSTNAME.example.org> <HOSTNAME> localhost

# End /etc/hosts (no network card version)
EOF
```

7.12. Crear enlaces simbólicos persnalizados a los dispositivos

7.12.1. Enlaces simbólicos a CD-ROM

Cierto software que puede que quieras instalar más adelante (por ejemplo, reproductores de medios) esperan que existan los enlaces simbólicos `/dev/cdrom` y `/dev/dvd` y que apunten a un dispositivo CD-ROOM o DVD-ROM. Igualmente, puede que te convenga poner referencias a dichos enlaces simbólicos en `/etc/fstab`. Udev incluye un guión que generará ficheros de reglas para crear dichos enlaces simbólicos por ti, dependiendo de las capacidades de cada dispositivo, pero deberás decidir cual de los dos modos de funcionamiento deseas que utilice el guión.

Primero, el guión puede operar en modo “by-path” (usado por defecto para dispositivos USB y FireWire), en el que las reglas creadas dependen de la ruta física del dispositivo CD o DVD. O puede operar en modo “by-id” (por defecto para dispositivos IDE y SCSI), en el que las reglas dependen de las cadenas de identificación almacenadas en el dispositivo CD o DVD. Las rutas son determinadas por el guión **path_id** de Udev, y las cadenas de identificación son leidas del hardware por sus programas **ata_id** o **scsi_id**, dependiendo del tipo de dispositivos que tengas.

Hay ventajas en cada método. El correcto a usar dependerá de qué tipo de cambios de dispositivos puedan ocurrir. Si esperas que cambie la ruta física a los dispositivos (esto es, los puertos y/o ranuras en los que estén conectados) debido por ejemplo a que planeas mover el controlador a un puerto IDE o USB diferente, entonces deberías usar el modo “by-id”. Por otro lado, si esperas que cambie la identificación del dispositivo, debido por ejemplo a una averia, y podrías sustituirlo por un dispositivo diferente pero con las mismas capacidades y conectado al mismo puerto, entonces deberías usar el modo “by-path”.

Si es posible que suceda cualquiera de estos tipos de cambio, entonces elige un modo basado el tipo de cambio que esperas suceda con mas frecuencia.

Importante

Los dispositivos externos (por ejemplo, un CD conectado por USB) no deberian usar el modo by-path, pues cada vez que se conecte a un nuevo puerto externo su ruta física cambiará. Todos los dispositivos conectados esternamente tendrán este problema si escribes reglas Udev para reconocerlos por su ruta física. El problema no se limita a dispositivos CD o DVD.

Si deseas ver los valores que los guiones Udev usarán, para el dispositivo CD-ROM apropiado encuentra el directorio correspondiente en `/sys` (puede ser, por ejemplo, `/sys/block/hdd`) y ejecuta un comando similar al siguiente:

```
udevtest /sys/block/hdd
```

Mira las lineas que contienen la salida de los diversos programas *_id. El modo “by-id” usará el valor ID_SERIAL si existe y no está vacio, en caso contrario usará una combinación de ID_MODEL y ID_REVISION. El modo “by-path” usa el valor ID_PATH.

Si el modo por defecto no es el adecuado para tu situación, puedes hacer la siguiente modificación en el fichero `/etc/udev/rules.d/75-cd-aliases-generator.rules` (donde *`mode`* es “by-id” o “by-path”):

```
sed -i -e 's/write_cd_rules/& mode/' \
    /etc/udev/rules.d/75-cd-aliases-generator.rules
```

Ten en cuenta que no es necesario crear los ficheros de reglas o enlaces simbolicos ahora, pues tienes montado el directorio `/dev` del anfitrión en el sistema LFS, y suponemos que los enlaces simbólicos existen en el anfitrión. Las reglas y enlaces simbólicos se crearán la primera vez que arranques tu sistema LFs.

Sin embargo, si tienes diversos dispositivos CD-ROM, los enlaces simbólicos generados en ese momento pueden apuntar a dispositivos diferentes de a los que apuntan en tu anfitrión, pues los dispositivos no se descubren en un orden predecible. Las asignaciones creadas cuando arrancas por primera vez el sistema LFS serán estables, por lo que esto solo es un problema si necesitas que los enlaces de ambos sistemas apunten a los mismos dispositivos. Si necesitas esto, revisa (y posiblemente edita) el fichero `/etc/udev/rules.d/70-persistent-cd.rules` generado tras el arranque para asegurate de que los enlaces cubren tus necesidades.

7.12.2. Manejar dispositivos duplicados

Como se explica en Sección 7.4, “Manejo de dispositivos y módulos en un sistema LFS”, el orden en el que dispositivos con la misma funcionalidad aparecen en `/dev` es aleatorio. Es decir, si tienes una cámara web USB y una sintonizadora de TV, a veces `/dev/video0` se refiere a la cámara y `/dev/video1` al sintonizador, pero tras un reinicio el orden puede ser el contrario. Para todas las clases de harware, excepto tarjetas de sonido y de red, esto puede solucionarse creando reglas udev para personalizar enlaces simbólicos persistentes. El caso de las tarjetas de red se cubre en Sección 7.13, “Configuración del guión network”, y la configuración de tarjetas de sonido se puede encontrar en *BLFS*.

Para cada uno de tus dispositivos que puedan tener este problema (incluso si el problema no existe en tu distribución Linux actual) encuentra el directorio correspondiente bajo `/sys/class` o `/sys/block`. Para dispositivos de vñideo este puede ser `/sys/class/video4linux/videoX`. Localiza los atributos que identifican inequívocamente al dispositivo (normalmente el fabricante, ID del producto o el número de serie):

```
udevinfo -a -p /sys/class/video4linux/video0
```

Entoces escribe reglas que creen los enlaces simbólicos:

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << EOF

# Enlaces persistentes para camaras web y sintonizadores
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", \
    SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f", ATTRS{vendor}=="0x109e", \
    SYMLINK+="tvtuner"

EOF
```

Nota

Ten en cuenta que Udev no reconoce la barra inversa como continuación de línea. Los ejemplos de este libro funcionan correctamente debido a que tanto la barra inversa como el salto de línea son ignorados por el interprete de comandos. Esto hace que el interprete de comandos envíe cada regla a cat como una única línea (el interprete de comandos ignora dicha secuencia debida a que la cadena EOF usada para redirigir el documento insertado no se encuentra delimitado con comillas simples o dobles. Para mas detalles consulta la página de manual bash(1) y busca "Here Documents").

Si modificas las reglas Udev con un editor de texto, asegurate de poner cada regla en una sola línea.

El resultado es que los dispositivos `/dev/video0` y `/dev/video1` aún se refieren de forma aleatoria al sintonizador o a la cámara web (y por tanto nunca deben usarse directamente) pero los enlaces simbólicos `/dev/tvtuner` y `/dev/webcam` siempre apuntarán al dispositivo correcto.

Puedes encontrar más información sobre la creación de reglas Udev en `/usr/share/doc/udev-113/index.html`.

7.13. Configuración del guión network

Esta sección solamente es aplicable en el caso de que vayas a configurar una tarjeta de red.

Si no tienes tarjeta de red es muy probable que no vayas a crear ninguna configuración relacionada con ellas. En ese caso, elimina los enlaces simbólicos `network` de todos los directorios de los niveles de ejecución (`/etc/rc.d/rc*.d`)

7.13.1. Crear nombres estables para las tarjetas de red

Con Udev y controladores de red modulares, la numeración de las interfaces de red no es persintente entre reinicios debido a que los controladores se cargan en paralelo, y por tanto en orden aleatorio. Por ejemplo, en un ordenador que tenga dos tarjetas de red fabricadas por Intel y Realtek la tarjeta de red fabricada por Intel podría ser `eth0` y la tarjeta Realtek ser `eth1`. En algunos casos, tras un reinicio las tarjetas podrían ser numeradas al contrario. Para evitar esto, Udev incluye un guión y algunas reglas para signar nombres estables a las tarjetas de red basados en sus direcciones MAC.

Pregenera las reglas para asegurar que los mismos nombres son asignados a los mismos dispositivos en cada arranque, incluido el primero:

```
/lib/udev/write_net_rules all_interfaces
```

Ahora inspecciona el fichero `/etc/udev/rules.d/70-persistent-net.rules` para saber qué nombre se le ha asignado a cada dispositivo de red:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

El fichero comienza con un comentario seguido por dos líneas para cada NIC. La primera línea de cada NIC es una descripción comentada que muestra sus ID de hardware (por ejemplo, si es una trajeta PCI, la ID del distribuidor PC y la ID del dispositivo), junto con sus controladores en parentesis, si puede encontrar el controlador. Ni el ID del hardware ni su controlador son usados para determinar qué nombre se le asigna a la interfaz. La segunda línea es la regla Udev para ese NIC que asigna su nombre.

Todas las reglas Udev están formadas por diferentes claves, separadas por comas y espacios opcionales. A continuación se muestran dichas claves de regla y su significado:

- `SUBSYSTEM=="net"` - Esto le indica a Udev que ignore los dispositivos que no sean tarjetas de red.
- `DRIVERS=="?*"` - Esto existe para que Udev ignore las subinterfaces VLAN o pasarela (pues estas subinterfaces no tienen controladores). Se ignoran pues el nombre que podría asignarseles podría colisionar con el de sus dispositivos padre.
- `ATTRS{type}=="1"` - Opcional. Esta clave se añadirá sólo si la NIC es un NIC wireless cuyo controlador crea múltiples interfaces virtuales. Esto asegura que la regla se aplique sólo a la interfaz primaria. Las interfaces secundarias se ignoran por la misma razón que las subinterfaces VLAN y pasarela: podría haver colisión de nombres.
- `ATTRS{address}` - El valor de esta clave es la dirección MAC de la NIC.
- `NAME` - El valor de esta clave es el nombre que Udev asignará a esta interface.

El valor de `NAME` es la parte importante. Asegurate de saber qué nombre ha sido asignado a cada una de tus tarjetas de red antes de continuar, y asegurate de usar dicho valor `NAME` cuando crees a continuación tus ficheros de configuración.

7.13.2. Creación de los ficheros de configuración de la interfaz de red

Qué interfaces de red activa o desactiva el guión network depende de los ficheros y directorios situados en el directorio `/etc/sysconfig/network-devices`. Este directorio debe contener un subdirectorio para cada interfaz a configurar, del tipo `ifconfig.xyz`, donde “xyz” es el nombre de una interfaz de red. Dentro de este directorio debería haber ficheros definiendo los atributos para dicha interfaz, como su direción(es) IP, mascaras de subred, etc..,

El siguiente comando crea un fichero `ipv4` de ejemplo para el dispositivo *eth0*:

```
cd /etc/sysconfig/network-devices
mkdir -v ifconfig.eth0
cat > ifconfig.eth0/ipv4 << "EOF"
ONBOOT=yes
SERVICE=ipv4-static
IP=192.168.1.1
GATEWAY=192.168.1.2
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Los valores de estas variables se deben cambiar en todos los ficheros por los valores apropiados. Si la variable `ONBOOT` tiene el valor “yes”, el guión network activará la NIC (Interfaz de Tarjeta de Red) correspondiente durante el arranque del sistema. Si contiene cualquier otro valor, el guión network ignorará la NIC correspondiente y no la activará.

La entrada `SERVICE` define el método usado para obtener la dirección IP. Los guiones de arranque de LFS tienen un formato de asignación de IP modular, y mediante la creación de ficheros adicionales en `/etc/sysconfig/network-devices/services` se permiten otros métodos de asignación IP. Esto se utiliza comúnmente para DHCP (Protocolo de Configuración Dinámica del Anfitrión), que se explica en el libro BLFS.

La variable `GATEWAY` debería contener la dirección IP de la puerta de enlace por efecto, si hay alguna. Si no, comenta la variable.

La variable PREFIX debe contener el número de bits usados en la subred. Cada octeto de una dirección IP tiene 8 bits. Si la máscara de subred es 255.255.255.0, estonces está usando los primeros tres octetos (24 bits) para especificar el número de red. Si la máscara de red es 255.255.255.240, podría estar usando los primeros 28 bits. Los prefijos mayores de 24 bits son usados normalmente por ISPs (Suministradores de Servicios de Internet) para DSL o cable. En este ejemplo (PREFIX=24), la máscara de red es 255.255.255.0. Ajústa la variable PREFIX de acuerdo a tu propia subred.

7.13.3. Creación del fichero /etc/resolv.conf

Si el sistema va a estar conectado a Internet, necesitará algún tipo de resolución de nombres DNS para resolver los nombres de dominio de Internet a direcciones IP y viceversa. Esto se consigue mejor colocando la dirección IP del servidor DNS, facilitado por el ISP o administrador de red, en /etc/resolv.conf. Crea este fichero ejecutando lo siguiente:

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

domain <Your Domain Name>
nameserver <IP address of your primary nameserver>
nameserver <IP address of your secondary nameserver>

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

Sustituye *<dirección IP del servidor de nombres>* con la dirección IP del servidor DNS más apropiado para tu configuración. Con frecuencia hay más de una entrada (los requisitos establecen servidores secundarios como respaldo). Si sólo necesitas o deseas un servidor DNS, elimina la segunda línea *nameserver* del fichero. La dirección IP pueder ser incluso un enrutador de la red local.

Capítulo 8. Hacer el sistema LFS arrancable

8.1. Introducción

Es hora de hacer arrancable el sistema LFS. En este capítulo se explica la creación de un fichero `fstab`, la construcción de un núcleo para el nuevo sistema LFS y la instalación del gestor de arranque GRUB para que el sistema LFS se pueda seleccionar para arrancar al inicio.

8.2. Creación del fichero /etc/fstab

El fichero `/etc/fstab` lo utilizan ciertos programas para determinar dónde deben montarse los sistemas de ficheros, en qué orden y cuales deben comprobarse (por fallos de integridad) antes de montarse. Crea una nueva tabla de sistemas de ficheros parecida a esta:

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

# file system  mount-point  type   options         dump  fsck
#                                                       order

/dev/<xxx>     /            <fff>  defaults        1     1
/dev/<yyy>     swap         swap   pri=1           0     0
proc           /proc        proc   defaults        0     0
sysfs          /sys         sysfs  defaults        0     0
devpts         /dev/pts     devpts gid=4,mode=620  0     0
shm            /dev/shm     tmpfs  defaults        0     0
# End /etc/fstab
EOF
```

Reemplaza *<xxx>*, *<yyy>* y *<fff>* con los valores apropiados para tu sistema, por ejemplo `hda2`, `hda5` y `ext3`. Para ver todos los detalles de los seis campos de este fichero, consulta **man 5 fstab**.

El punto de montaje `/dev/shm` para `tmpfs` se incluye para permitir la activación de la memoria compartida POSIX. Tu núcleo debe tener compilado en su interior el soporte requerido para que funcione (más datos sobre esto en la siguiente sección). Ten en cuenta que actualmente muy poco software utiliza en realidad la memoria compartida POSIX. Por tanto, puedes considerar como opcional el montaje de `/dev/shm`. Para más información consulta `Documentation/filesystems/tmpfs.txt` en el árbol de fuentes del núcleo.

Los sistemas de ficheros originados en MS-DOS o Windows (vfat, ntfs, smbfs, cifs, iso9660, udf) necesitan la opción de montaje “iocharset” para poder interpretar correctamente los caractéres no ASCII en los nombres de ficheros. El valor de esta opción debería ser el mismo del grupo de caractéres de tu locale, ajustado de forma que el núcleo pueda entenderlo. Esto funciona si la definición del grupo de caractéres apropiada (que se encuentra bajo Sistemas de ficheros -> Soporte para Lenguaje Nativo) ha sido compilada dentro del núcleo o como módulo. La opción “codepage”también es necesaria para los sistemas de ficheros vfat y smbfs. Debería establecerse al número de código de página usado en tu país bajo MS-DOS. Por ejemplo, un usuario ru_RU.KOI8-R podría necesitar lo siguiente en la parte de opciones de su línea de montaje en `/etc/fstab`:

```
noauto,user,quiet,showexec,iocharset=koi8r,codepage=866
```

Las opciones correspondiente para usuarios ru_RU.UTF-8 es:

```
noauto,user,quiet,showexec,iocharset=utf8,codepage=866
```

Nota

En el último caso el núcleo emitirá el siguiente mensaje:

```
FAT: utf8 is not a recommended IO charset for FAT filesystems,
    filesystem will be case sensitive!
```

Esta recomendación negativa debería ignorarse, pues todos los demas valores de la opción “iocharset” producen resultados erroneos en pantalla para los nombres de fichero en locales UTF-8.

También es posible especificar los valores del código de página y del grupo de caracteres de IO por defecto para algunos sistemas de ficheros durente la configuración del núcleo. Los parámetros relacionados son: “Default NLS Option” (`CONFIG_NLS_DEFAULT`), “Default Remote NLS Option” (`CONFIG_SMB_NLS_DEFAULT`), “Default codepage for FAT” (`CONFIG_FAT_DEFAULT_CODEPAGE`), y “Default iocharset for FAT” (`CONFIG_FAT_DEFAULT_IOCHARSET`). No hay forma de configurar estos ajustes para el sistema de ficheros ntfs durante la compilación del núcleo.

8.3. Linux-2.6.22.6

El paquete Linux contiene el núcleo Linux.

Tiempo estimado de construcción: 1.5 - 3.5 SBU
Espacio requerido en disco: 350 - 450 MB

8.3.1. Instalación del núcleo

Construir el núcleo comprende varios pasos: configuración, compilación e instalación. Mira en el fichero `README` del árbol de fuentes del núcleo los métodos de configuración del núcleo alternativos al utilizado en este libro.

Prepara la compilación ejecutando el siguiente comando:

```
make mrproper
```

Esto asegura que el árbol del núcleo está completamente limpio. El equipo del núcleo recomienda que se ejecute este comando antes de cada compilación del núcleo. No debes confiar en que el árbol de las fuentes esté limpio tras desempaquetarlo.

Configura el núcleo mediante una interfaz de menús. BLFS tiene información sobre requisitos particulares de configuraciones del núcleo para paquetes externos a LFS en *http://www.linuxfromscratch.org/blfs/view/svn/longindex.html#kernel-config-index*:

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

Significado de los parámetros de make:

`LANG=<host_LANG_value> LC_ALL=`

Esto establece los ajustes de locale a aquellos usados en el anfitrión. Esto es necesario para un correcto dibujo de líneas de la interface ncurses de menuconfig en consolas linux de texto basadas en UTF-8.

Asegurate de reemplazar *`<host_LANG_value>`* por el valor de la variable `$LANG` de tu anfitión. Si esta no está establecida, puedes usar en su lugar el valor en el anfitrión de `$LC_ALL` o `$LC_CTYPE`.

Alternativamente, **`make oldconfig`** puede ser más adecuado en algunas situaciones. Lee el fichero `README` para más detalles.

Si lo deseas, sáltate la configuración del núcleo copiando el fichero de configuración del núcleo, `.config`, de tu sistema anfitrión (asumiendo que esté disponible) al directorio `linux-2.6.22.6`. Sin embargo, no recomendamos esta opción. Con frecuencia es mejor explorar todos los menús de configuración y crear tu propia configuración del núcleo desde cero.

Compila la imagen del núcleo y los módulos:

```
make
```

Si utilizas los módulos del núcleo puede que necesites un fichero `/etc/modprobe.conf`. La información relativa a los módulos y a la configuración del núcleo se encuentra en Sección 7.4, “Manejo de dispositivos y módulos en un sistema LFS” y en el directorio `linux-2.6.22.6/Documentation`, que contiene la documentación del núcleo. Igualmente, `modprobe.conf(5)` puede ser interesante.

Instala los módulos, si la configuración del núcleo los utiliza:

```
make modules_install
```

Tras completar la compilación se necesitan algunos pasos adicionales para completar la instalación. Es necesario copiar varios ficheros al directorio `/boot`.

La ruta a la imagen del núcleo puede variar dependiendo de la plataforma que utilices. El siguiente comando asume que la arquitectura es x86:

```
cp -v arch/i386/boot/bzImage /boot/lfskernel-2.6.22.6
```

`System.map` es un fichero de símbolos para el núcleo. Mapea los puntos de entrada de cada una de las funciones en la API del núcleo, al igual que las direcciones de las estructuras de datos del núcleo para el núcleo en ejecución. Ejecuta el siguiente comando para instalar el fichero de mapa:

```
cp -v System.map /boot/System.map-2.6.22.6
```

`.config` es el fichero de configuración del núcleo creado por el paso **make menuconfig** anterior. Contiene todas las selecciones de configuración para el núcleo que se acaba de compilar. Es buena idea guardar este fichero como referencia futura:

```
cp -v .config /boot/config-2.6.22.6
```

Instala la documentación del núcleo Linux:

```
install -d /usr/share/doc/linux-2.6.22.6
cp -r Documentation/* /usr/share/doc/linux-2.6.22.6
```

Es importante advertir que los ficheros del directorio de las fuentes del núcleo no son propiedad de *root*. Cuando se desempaqueta un paquete como usuario *root* (como hacemos dentro del chroot), los ficheros acaban teniendo los identificadores de usuario y grupo que tenían en la máquina en la que se empaquetaron. Esto normalmente no es problema para cualquier otro paquete que instales debido a que eliminas las fuentes tras la instalación. Pero con frecuencia el árbol de las fuentes de Linux se guarda durante mucho tiempo, por lo que es posible que el ID de usuario del empaquetador sea asignado a alguien en tu máquina y entonces dicha persona podría tener permiso de escritura en las fuentes del núcleo.

Si vas a guardar el árbol de las fuentes del núcleo, ejecuta **`chown -R 0:0`** sobre el directorio `linux-2.6.22.6` para asegurar que todos los ficheros son propiedad de *root*.

Aviso

Cierta documentación del núcleo recomienda crear un enlace simbólico `/usr/src/linux` que apunte al directorio de las fuentes del núcleo. Esto es específico para núcleos anteriores a la serie 2.6 y *no debe* ser creado en un sistema LFS, pues puede causar problemas con los paquetes que desees instalar una vez que tu sistema LFS esté completo.

Aviso

Las cabeceras del directorio `include` del sistema deben ser *siempre* aquellas contra las que se compiló Glibc, es decir, las cabeceras saneadas procedentes de este paquete del núcleo Linux. Por tanto *nunca* deben reemplazarse con las cabeceras crudas del núcleo ni con las cabeceras saneadas de otro núcleo.

8.3.2. Contenido de Linux

Ficheros instalados: config-2.6.22.6, lfskernel-2.6.22.6, and System.map-2.6.22.6

Descripciones cortas

`config-2.6.22.6`	Contiene todas las opciones de configuración del núcleo.
`lfskernel-2.6.22.6`	El corazón del sistema GNU/Linux. Cuando enciendes tu ordenador, lo primero que se carga del sistema operativo es el núcleo. Éste detecta e inicializa todos los componentes hardware del ordenador, poniendo estos componentes a disposición del software como si fuesen un árbol de ficheros y convierte una CPU única en una máquina multi-tarea capaz de ejecutar concurrentemente varios programas casi al mismo tiempo.
`System.map-2.6.22.6`	Un listado de direcciones y símbolos. Mapea los puntos de entrada y direcciones de todas las funciones y estructuras de datos del núcleo.

8.4. Hacer el sistema LFS arrancable

Tu nuevo y brillante sistema LFS está casi completo. Una de las últimas cosas por hacer es asegurarse de que puede ser arrancado. Las siguientes instrucciones sólo son aplicables en ordenadores de arquitectura IA-32, o sea PCs. La información sobre "cargadores de arranque" para otras arquitecturas debería estar disponible en las localizaciones usuales de recursos específicos para esas arquitecturas.

El arranque puede ser una tarea compleja. Primero, unas palabras de advertencia. Familiarízate con tu actual gestor de arranque y con cualquier otro sistema operativo presente en tu(s) disco(s) duro(s) que desees mantener arrancable. Asegúrate de que tienes preparado un disco de arranque de emergencia para poder "rescatar" el ordenador si este quedase inutilizable (no arrancable).

Anteriormente compilamos e instalamos el gestor de arranque GRUB en preparación para este paso. El proceso consiste en escribir ciertos ficheros especiales de GRUB a su localización específica en el disco duro. Antes de hacer esto te recomendamos encarecidamente que crees un disquete de arranque de GRUB como respaldo. Inserta un disquete en blanco y ejecuta los siguientes comandos:

```
dd if=/boot/grub/stage1 of=/dev/fd0 bs=512 count=1
dd if=/boot/grub/stage2 of=/dev/fd0 bs=512 seek=1
```

Saca el disquete y guárdalo en lugar seguro. Ahora inicia el intérprete de comandos de **grub**:

```
grub
```

GRUB utiliza su propia estructura de nombres para los discos de la forma *(hdn,m)*, donde *n* es el número del disco duro y *m* es el número de la partición, comenzando ambos desde 0. Por ejemplo, la partición `hda1` es *(hd0,0)* para GRUB, y `hdb3` es *(hd1,2)*. Al contrario que Linux, GRUB no considera los dispositivos CD-ROM como discos duros. Por ejemplo, si tienes un CD en `hdb` y un segundo disco duro en `hdc`, este segundo disco duro seguiría siendo *(hd1)*.

Usando la información anterior, determina la denominación apropiada para tu partición raíz (o partición de arranque, si usas una separada). Para los siguientes ejemplos asumiremos que tu partición raíz (o la de arranque) es `hda4`

Indícale a GRUB dónde debe buscar sus ficheros `stage{1,2}`. Puedes utilizar el tabulador para que GRUB te muestre las alternativas:

```
root (hd0,3)
```

Aviso

El siguiente comando sobreescribirá tu actual gestor de arranque. No ejecutes el comando si esto no es lo que quieres. Por ejemplo, si utilizas otro gestor de arranque para administrar tu MBR (Master Boot Record, Registro Maestro de Arranque). En este escenario, posiblemente tenga más sentido instalar GRUB en el "sector de arranque" de la partición LFS, en cuyo caso dicho comando sería **`setup (hd0,3)`**.

Indícale a GRUB que se instale en el MBR de `hda`:

```
setup (hd0)
```

Si todo está bien, GRUB informará que ha encontrado sus ficheros en `/boot/grub`. Esto es todo para activarlo. Cierra el intérprete de comandos de **grub**:

```
quit
```

Crea un fichero de “lista de menú” para definir el menú de arranque de GRUB:

```
cat > /boot/grub/menu.lst << "EOF"
# Begin /boot/grub/menu.lst

# By default boot the first menu entry.
default 0

# Allow 30 seconds before booting the default.
timeout 30

# Use prettier colors.
color green/black light-green/black

# The first entry is for LFS.
title LFS SVN-20070916
root (hd0,3)
kernel /boot/lfskernel-2.6.22.6 root=/dev/hda4
EOF
```

Si lo desas, añade una entrada para la distribución anfitriona. Tendrá un aspecto similar a este:

```
cat >> /boot/grub/menu.lst << "EOF"
title Red Hat
root (hd0,2)
kernel /boot/kernel-2.6.5 root=/dev/hda3
initrd /boot/initrd-2.6.5
EOF
```

Si necesitas un arranque dual a Windows, la siguiente entrada debería permitirte iniciarlo:

```
cat >> /boot/grub/menu.lst << "EOF"
title Windows
rootnoverify (hd0,0)
chainloader +1
EOF
```

Si **info grub** no te dice todo lo que quieres saber, puedes encontrar más información sobre GRUB en su sitio web, localizado en: *http://www.gnu.org/software/grub/*.

El estándar FHS estipula que el fichero `menu.lst` debe tener un enlace simbólico a `/etc/grub/menu.lst`. Para satisfacer este requisito, ejecuta el siguiente comando:

```
mkdir -v /etc/grub
ln -sv /boot/grub/menu.lst /etc/grub
```

Capítulo 9. El final

9.1. El final

¡Bien hecho! ¡El nuevo sistema LFS está instalado! Te deseamos mucha diversión con tu flamante sistema Linux hecho a la medida.

Puede ser una buena idea crear un fichero `/etc/lfs-release`. Teniendo este fichero te será muy fácil (y a nosotros, si es que vas a pedir ayuda en algún momento) saber qué versión de LFS tienes instalada en tu sistema. Crea este fichero ejecutando:

```
echo SVN-20070916 > /etc/lfs-release
```

9.2. Registrarse

Ahora que has terminado el libro, ¿qué te parecería poder registrarte como usuario de LFS? Visita *http://www.linuxfromscratch.org/cgi-bin/lfscounter.cgi* y regístrate como usuario de LFS introduciendo tu nombre y la primera versión de LFS que has usado.

Arranquemos ahora el sistema LFS.

9.3. Reinicio del sistema

Ahora que se han instalado todos los programas, es hora de reiniciar el ordenador. Sin embargo, debes tener en cuenta varias cosas. El sistema que has creado en este libro es bastante reducido y muy posiblemente no tenga la funcionalidad que podrías necesitar para seguir adelante. Instalar varios paquetes adicionales del libro BLFS mientras aún estás en el entorno chroot te dejará en una mejor posición para continuar una vez que reinicies tu nueva instalación LFS. Al instalar un navegador web en modo texto, como Lynx, podrás facilmente ver el libro BLFS en una terminal mientras compilas los paquetes en otra. El paquete GPM te permitirá copìar y pegar en tu terminal virtual. Por último, si estás en una situación en la que una configuración de IP estática no cubre los requisitos de tu red, instalar ahora paquetes como Dhcpcd o PPP es también útil.

Una vez dicho esto, ¡vayamos a arrancar nuestra nueva instalación de LFS por primera vez!. Primero sal del entorno chroot:

```
logout
```

Desmonta los sistemas de ficheros virtuales:

```
umount -v $LFS/dev/pts
umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/proc
umount -v $LFS/sys
```

Desmonta el sistema de ficheros del LFS:

```
umount -v $LFS
```

Si creaste varias particiones, desmonta las otras particiones antes de desmontar la principal, por ejemplo:

```
umount -v $LFS/usr
umount -v $LFS/home
umount -v $LFS
```

Ahora reinicia el sistema con:

```
shutdown -r now
```

Asumiendo que el gestor de arranque GRUB fue configurado como se indicó anteriormente, el menú está establecido para que *LFS SVN-20070916* arranque automáticamente.

Una vez terminado el reinicio, el sistema LFS está listo para su uso y puedes añadir más software para cubrir tus necesidades.

9.4. ¿Y ahora, qué?

Gracias por leer el libro LFS. Esperamos que lo hayas encontrado útil y hayas aprendido algo sobre el proceso de creación del sistema.

Ahora que el sistema LFS está instalado, puede que te preguntes “¿Y ahora, qué?”. Para responder esta cuestión te hemos preparado una lista de recursos.

- Mantenimiento

  Con regularidad se hacen informes con los errores y fallos de seguridad para todo el software. Puesto que el sistema LFS se compila desde las fuentes, eres tú quien debe estar al tanto de dichos informes. Hay diversos recursos en línea para monitorizar dichos informes. A continuación se muestran algunos de ellos:

  - Freshmeat.net (*http://freshmeat.net/*)

    Freshmeat puede notificarte (por correo electrónico) de las nuevas versiones de los paquetes instalados en tu sistema.

  - *CERT* (Computer Emergency Response Team)

    CERT tiene una lista de correo en la que publica alertas de seguridad concernientes a varios sistemas operativos y aplicaciones. La información para subscribirse está disponible en *http://www.us-cert.gov/cas/signup.html*.

  - Bugtraq

    Bugtraq es una lista de correo de acceso total sobre seguridad en ordenadores. Publica los problemas de seguridad recién descubiertos y, ocasionalmente, sus posibles correcciones. La información para subscribirse está disponible en *http://www.securityfocus.com/archive*.

- Más Allá de Linux From Scratch

  El libro Más Allá de Linux From Scratch (BLFS) cubre los procesos de instalación de paquetes muy diferentes que están fuera del objetivo del Libro LFS. Puedes encontrar el proyecto BLFS en *http://www.linuxfromscratch.org/blfs/*.

- Recetas de LFS

  Las recetas de LFS son una serie de documentos educacionales, suministrados por voluntarios a la comunidad LFS. Las recetas están disponibles en *http://www.linuxfromscratch.org/hints/list.html*.

- Listas de Correo

  Hay varias listas de correo sobre LFS a las que puedes suscribirte si necesitas ayuda, si quieres mantenerte al corriente de los últimos desarrollos, si quieres contribuir al proyecto y más. Para más información consulta el Capítulo 1 - Listas de correo.

- El Proyecto de Documentación de Linux (TLDP)

  El objetivo del Proyecto de Documentación de Linux es colaborar en todo lo relacionado con la creación y publicación de la documentación sobre Linux. El LDP ofrece una gran colección de CÓMOS, Guías y páginas de manual y puedes encontrarlo en *http://www.tldp.org/*. Su filial en castellano se encuentra en *http://es.tldp.org*.

Parte IV. Apéndices

Apéndice A. Acrónimos y términos

ABI Application Binary Interface - Interfaz de Aplicación Binaria

ALFS Automated Linux From Scratch - Linux From Scratch Automatizado

ALSA Advanced Linux Sound Architecture - Arquitectura Avanzada de Sonido en Linux

API Application Programming Interface - Interfaz de Aplicación para Programación

ASCII American Standard Code for Information Interchange - Código Americano Estándar para el Intercambio de Información

BIOS Basic Input/Output System - Sistema Básico de Entrada/Salida

BLFS Beyond Linux From Scratch - Más Allá de Linux From Scratch

BSD Berkeley Software Distribution - Distribución Berkeley de Software

chroot change root - cambio de raíz

CMOS Complementary Metal Oxide Semiconductor - Semiconductor Complementario de Oxido de Metal

COS Class Of Service - Clase De Servicio

CPU Central Processing Unit - Unidad Central de Procesamiento

CRC Cyclic Redundancy Check - Comprobación Cíclica de Redundancia

CVS Concurrent Versions System - Sistema Concurrente de Versiones

DHCP Dynamic Host Configuration Protocol - Protocolo Dinámico de Configuración del Anfitrión

DNS Domain Name Service - Servicio de Nombres de Dominio

EGA Enhanced Graphics Adapter - Adaptador Mejorado de Gráficos

ELF Executable and Linkable Format - Formato Ejecutable y Enlazable

EOF End Of File - Fin De Fichero

EQN equation - ecuación

EVMS Enterprise Volume Management System - Sistema Empresarial de Administración de Volúmenes

ext2 second extended file system - segundo sistema de ficheros extendido

ext3 third extended file system - tercer sistema de ficheros extendido

FAQ Frequently Asked Questions - Cuestiones Preguntadas Frecuentemente

FHS Filesystem Hierarchy Standard - Estándar de la Jerarquía de Sistemas de Ficheros

FIFO First In, First Out - Primero en Entrar, Primero en Salir

FQDN Fully Qualified Domain Name - Nombre de Dominio Completamente Cualificado

FTP File Transfer Protocol - Protocolo de Transferencia de Ficheros

GB Gibabytes

GCC GNU Compiler Collection - Colección GNU de Compiladores

GID Group Identifier - Identificador de Grupo

GMT Greenwich Mean Time - Tiempo del Meridiano de Greenwich

GPG GNU Privacy Guard - Guardián GNU de Privacidad

HTML Hypertext Markup Language - Lenguaje de Marcas de Hipertexto
IDE Integrated Drive Electronics - Controlador Electrónico Integrado
IEEE Institute of Electrical and Electronic Engineers - Instituto de Ingenieros en Electricidad y Electrónica
IO Input/Output - Entrada/Salida
IP Internet Protocol - Protocolo de Internet
IPC Inter-Process Communication - Comunicación Entre Procesos
IRC Internet Relay Chat - Charlas en Internet
ISO International Organization for Standardization - Organización Internacional para la Estandarización
ISP Internet Service Provider - Proveedor de Servicios de Internet
KB Kilobytes
LED Light Emitting Diode - Diodo Emisor de Luz
LFS Linux From Scratch
LSB Linux Standard Base - Estándar Base de Linux
MB Megabytes
MBR Master Boot Record - Registro Maestro de Arranque
MD5 Message Digest 5 - Resúmen 5 del Mensaje
NIC Network Interface Card - Tarjeta de Interfaz de Red
NLS Native Language Support - Soporte para Lenguaje Nativo
NNTP Network News Transport Protocol - Protocolo de Red para Transporte de Noticias
NPTL Native POSIX Threading Library - Librería Nativa de Hilos POSIX
OSS Open Sound System - Sistema Abierto de Sonido
PCH Pre-Compiled Headers - Cabeceras Precompiladas
PCRE Perl Compatible Regular Expression - Expresión Regular Compatible de Perl
PID Process Identifier - Identificador del Proceso
PLFS Pure Linux From Scratch - Linux From Scratch Puro
PTY pseudo terminal
QA Quality Assurance - Control de Calidad
QOS Quality Of Service - Calidad Del Servicio
RAM Random Access Memory - Memoria de Acceso Aleatorio
RPC Remote Procedure Call - Llamada a Procedimiento Remoto
RTC Real Time Clock - Reloj de Tiempo Real
SBU Standard Build Unit - Unidad Estándar de Construcción
SCO The Santa Cruz Operation
SGR Select Graphic Rendition - Interpretación de la Selección Gráfica
SHA1 Secure-Hash Algorithm 1 - Algoritmo 1 de Tabla Segura
SMP Symmetric Multi-Processor - Multiprocesador Simétrico

TLDP The Linux Documentation Project - El Proyecto de Documentación Linux

TFTP Trivial File Transfer Protocol - Protocolo Trivial de Transferencia de Ficheros

TLS Thread-Local Storage - Almacenamiento Local de Hilos

UID User Identifier - Identificador de Usuario

umask user file-creation mask - máscara de creación de ficheros del usuario

USB Universal Serial Bus - Bus Serie Universal

UTC Coordinated Universal Time - Tiempo Universal Coordinado

UUID Universally Unique Identifier - Identificador Universalmente Unico

VC Virtual Console - Consola Virtual

VGA Video Graphics Array - Matríz de Gráficos de Vídeo

VT Virtual Terminal - Terminal Virtual

Apéndice B. Agradecimientos

Queremos agradecer a las siguientes personas y organizaciones su contribución al Proyecto LFS-ES:

- *Gerard Beekmans*, por crear el apasionante proyecto Linux From Scratch.
- *Red ECOLNET*, por prestarnos su apoyo incondicional desde el primer momento y facilitarnos los servicios de SVN, listas de correo y espacio web, que son vitales para realizar nuestro trabajo.
- *Alberto Ferrer*, por poner a nuestra disposición los servicios de hospedaje de Dattatec.
- *Al Equipo del LFS-ES*, por su dedicación e interés en conseguir que este proyecto funcione y que las traducciones tengan la mejor calidad posible.
- A todos aquellos que leen nuestras traducciones con interés, pues es para ellos para quienes las escribimos.

Queremos agradecer a las siguientes personas y organizaciones su contribución al Proyecto Linux From Scratch:

- *Gerard Beekmans* <gerard@linuxfromscratch.org> – Creador de LFS, lider del Proyecto LFS.
- *Matthew Burgess* <matthew@linuxfromscratch.org> – Líder del proyecto LFS, escritor/editor técnico de LFS.
- *Bruce Dubbs* <bdubbs@linuxfromscratch.org> – Administrador de la publicación de LFS.
- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – Mantenedor de los XML y XSL de LFS/BLFS/HLFS.
- *Jim Gifford* <jim@linuxfromscratch.org> – Co-lider del proyecto CLFS.
- *Bryan Kadzban* <bryan@linuxfromscratch.org> – Escritor técnico de LFS.
- *Jeremy Huntwork* <jhuntwork@linuxfromscratch.org> – Escritor técnico de LFS, mantenedor del LiveCD de LFS.
- *Randy McMurchy* <randy@linuxfromscratch.org> – Lider del proyecto BLFS.
- *Dan Nicholson* <dnicholson@linuxfromscratch.org> – Editor de LFS y BLFS
- *Ken Moffat* <ken@linuxfromscratch.org> – Editor de LFS y CLFS.
- *Ryan Oliver* <ryan@linuxfromscratch.org> – Co-lider del proyecto CLFS
- *Alexander E. Patrakov* <alexander@linuxfromscratch.org> – Escritor técnico de LFS, editor de la intenacionalización de LFS, mantenedor del LiveCD de LFS.
- Innumerables personas de las diversas listas de correo de LFS y BLFS que han hecho que este libro sea posible mediante sus sugerencias, probando el libro y suministrando informes de errores, instrucciones y sus experiencias con la instalación de diversos paquetes.

Traductores

- *Manuel Canales Esparcia* <macana@macana-es.com> – Proyecto de traducción al castellano de LFS.
- *Johan Lenglet* <johan@linuxfromscratch.org> – Proyecto de traducción al francés de LFS.
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Proyecto de traducción al portugués de LFS.
- *Thomas Reitelbach* <tr@erdfunkstelle.de> – Proyecto de traducción al alemán de LFS.

Administradores de la red de réplicas

América del Norte

- *Scott Kveton* <scott@osuosl.org> – lfs.oregonstate.edu.

- *William Astle* <lost@l-w.net> – ca.linuxfromscratch.org.
- *Eujon Sellers* <jpolen@rackspace.com> – lfs.introspeed.com.
- *Justin Knierim* <tim@idge.net> – lfs-matrix.net.

América del Sur

- *Manuel Canales Esparcia* <manuel@linuxfromscratch.org> – lfsmirror.lfs-es.info.
- *Luis Falcon* <Luis Falcon> – torredehanoi.org.

Europa

- *Guido Passet* <guido@primerelay.net> – nl.linuxfromscratch.org.
- *Bastiaan Jacques* <baafie@planet.nl> – lfs.pagefault.net.
- *Sven Cranshoff* <sven.cranshoff@lineo.be> – lfs.lineo.be.
- Scarlet Belgium – lfs.scarlet.be.
- *Sebastian Faulborn* <info@aliensoft.org> – lfs.aliensoft.org.
- *Stuart Fox* <stuart@dontuse.ms> – lfs.dontuse.ms.
- *Ralf Uhlemann* <admin@realhost.de> – lfs.oss-mirror.org.
- *Antonin Sprinzl* <Antonin.Sprinzl@tuwien.ac.at> – at.linuxfromscratch.org.
- *Fredrik Danerklint* <fredan-lfs@fredan.org> – se.linuxfromscratch.org.
- *Franck* <franck@linuxpourtous.com> – lfs.linuxpourtous.com.
- *Philippe Baqué* <baque@cict.fr> – lfs.cict.fr.
- *Vitaly Chekasin* <gyouja@pilgrims.ru> – lfs.pilgrims.ru.
- *Benjamin Heil* <kontakt@wankoo.org> – lfs.wankoo.org.

Asia

- *Satit Phermsawang* <satit@wbac.ac.th> – lfs.phayoune.org.
- *Shizunet Co.,Ltd.* <info@shizu-net.jp> – lfs.mirror.shizu-net.jp.
- *Init World* <http://www.initworld.com/> – lfs.initworld.com.

Australia

- *Jason Andrade* <jason@dstc.edu.au> – au.linuxfromscratch.org.

Anteriores miembros de los equipos

- *Christine Barczak* <theladyskye@linuxfromscratch.org> – Editora del libro LFS.
- Archaic <archaic@linuxfromscratch.org> – Editor y escritor técnico de LFS, editor de BLFS, mantenedor de los proyectos Hints y Patches.
- *Nathan Coulson* <nathan@linuxfromscratch.org> – Mantenedor de LFS-Bootscripts.
- Timothy Bauscher
- Robert Briggs

- Ian Chilton
- *Jeroen Coumans* <jeroen@linuxfromscratch.org> – Desarrollador del sitio web, mantenedor de la FAQ.
- Alex Groenewoud – Escritor técnico de LFS
- Marc Heerdink
- Mark Hymers
- Seth W. Klein – Mantenedor de las FAQ
- *Nicholas Leippe* <nicholas@linuxfromscratch.org> – Mantenedor del Wiki.
- *Anderson Lizardo* <lizardo@linuxfromscratch.org> – Mantenedor de los guiones de respaldo del sitio web.
- Simon Perreault
- *Scot Mc Pherson* <scot@linuxfromscratch.org> – Mantenedor de la pasarela NNTP de LFS.
- *Greg Schafer* <gschafer@zip.com.au> – Escritor técnico de LFS.
- Jesse Tie-Ten-Quee – Escritor técnico de LFS.
- *James Robertson* <jwrober@linuxfromscratch.org> – Mamtenedor de Bugzilla.
- *Tushar Teredesai* <tushar@linuxfromscratch.org> – Editor de BLFS, lider de los proyectos Hints y Patches.
- *Jeremy Utley* <jeremy@linuxfromscratch.org> – Escritor técnico de LFS, mantenedor del Bugzilla, mantenedor de LFS-Bootscripts.
- *Zack Winkles* <zwinkles@gmail.com> – Escritor técnico de LFS.

Un agradecimiento muy especial a nuestros donadores

- *Dean Benson* <dean@vipersoft.co.uk> por múltiples donaciones monetarias.
- *Hagen Herrschaft* <hrx@hrxnet.de> por donar un sistema P4 a 2.2GHz, que ahora corre bajo el nombre de Lorien.
- *SEO Company Canada* apoya a proyectos de código abierto y a diversas distribuciones Linux
- *VA Software* que, en nombre de *Linux.com*, donó una estación de trabajo VA Linux 420 (antes StartX SP2).
- Mark Stone por donar Belgarath, el primer servidor de linuxfromscratch.org.

Apéndice C. Dependencias

Todo paquete construido en LFS depende de otros paquetes para poder construirse e instalarse correctamente. Algunos paquetes incluso participan en dependencias circulares, esto es, el primer paquete depende del segudo que a su vez depende del primero. Debido a estas dependencias, el orden en el cual se construyen los paquetes de LFS es muy importante. El proposito de esta página es documentar las dependencias de cada paquete construido en LFS.

Para cada paquete que construimos, tenemos listadas tres tipos de dependencias. La primera lista qué otros paquetes deben estar disponibles para compilar e instalar el paquete en cuestion. La segunda lista qué paquetes, en adición a los de la primera lista, deben estar disponibles para ejecutar los bancos de pruebas. La última lista de dependencias son paquetes que necesitan que este paquete sea construido e instalado en su localización final antes de que ellos sean construidos e instalados. En muchos casos, esto se debe a que dichos paquetes incluirán rutas completas a los binarios dentro de sus guiones. Si no se construyen en un cierto orden, esto podría provocar que rutas del tipo /tools/bin/[binario] sean añadidas dentro de los guiones instalados en el sistema final. Esto, obviamente, no es deseable.

Autoconf

Para su instalación depende de:	Bash, Coreutils, Grep, M4, Make, Perl, Sed y Texinfo
El banco de pruebas depende de:	Automake, Diffutils, Findutils, GCC y Libtool
Debe instalarse antes de:	Automake

Automake

Para su instalación depende de:	Autoconf, Bash, Coreutils, Gettext, Grep, M4, Make, Perl, Sed y Texinfo
El banco de pruebas depende de:	Binutils, Bison, Bzip2, DejaGNU, Diffutils, Expect, Findutils, Flex, GCC, Gettext, Gzip, Libtool y Tar. También puede usar otros paquetes no instalados en LFS.
Debe instalarse antes de:	Ninguno

Bash

Para su instalación depende de:	Bash, Bison, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Readline, Sed y Texinfo
El banco de pruebas depende de:	Diffutils y Gawk
Debe instalarse antes de:	Ninguno

Berkeley DB

Para su instalación depende de:	Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make y Sed
El banco de pruebas depende de:	No se ejecuta. Necesita que TCL sea instalado en el sistema final.
Debe instalarse antes de:	Ninguno

Binutils

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Perl, Sed y Texinfo

El banco de pruebas depende de: DejaGNU y Expect

Debe instalarse antes de: Ninguno

Bison

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make y Sed

El banco de pruebas depende de: Diffutils y Findutils

Debe instalarse antes de: Flex, Kbd y Tar

Bzip2

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Make y Patch

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

Coreutils

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed y Texinfo

El banco de pruebas depende de: Diffutils, E2fsprogs

Debe instalarse antes de: Bash, Diffutils, Findutils, Man-DB y Udev

DejaGNU

Para su instalación depende de: Bash, Coreutils, Diffutils, GCC, Grep, Make y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Diffutils

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Expect

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed y Tcl

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

E2fsprogs

Para su instalación depende de: Bash, Binutils, Coreutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, Make, Sed y Texinfo

El banco de pruebas depende de: Diffutils

Debe instalarse antes de: Util-Linux

File

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed y Zlib

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Findutils

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed y Texinfo

El banco de pruebas depende de: DejaGNU, Diffutils y Expect

Debe instalarse antes de: Ninguno

Flex

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, M4, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: Bison y Gawk

Debe instalarse antes de: IPRoute2, Kbd y Man-DB

Gawk

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: Diffutils

Debe instalarse antes de: Ninguno

Gcc

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make, Patch, Perl, Sed, Tar y Texinfo

El banco de pruebas depende de: DejaGNU y Expect

Debe instalarse antes de: Ninguno

Gettext

Para su instalación depende de: Bash, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Make, Sed y Texinfo

El banco de pruebas depende de: Diffutils, Perl y Tcl

Debe instalarse antes de: Automake

Glibc

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Gettext, Grep, Gzip, Make, Perl, Sed, and Texinfo

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

Grep

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: Diffutils y Gawk

Debe instalarse antes de: Man-DB

Groff

Para su instalación depende de: Bash, Binutils, Bison, Coreutils, Gawk, GCC, Glibc, Grep, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Man-DB y Perl

GRUB

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses, Sed y Texinfo

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

Gzip

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Man-DB

Iana-Etc

Para su instalación depende de: Coreutils, Gawk y Make

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Perl

Inetutils

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed y Texinfo

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Tar

IProute2

Para su instalación depende de: Bash, Berkeley DB, Bison, Coreutils, Flex, GCC, Glibc, Make y las cabeceras API de Linux

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Ninguno

Kbd

Para su instalación depende de: Bash, Binutils, Bison, Coreutils, Flex, GCC, Gettext, Glibc, Gzip, Make, Patch y Sed

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Ninguno

Less

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses y Sed

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Ninguno

Libtool

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Sed y Texinfo

El banco de pruebas depende de: Findutils

Debe instalarse antes de: Ninguno

Linux Kernel

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Findutils, GCC, Glibc, Grep, Gzip, Make, Module-Init-Tools, Ncurses y Sed

El banco de pruebas depende de: No incluye un bamco de pruebas.

Debe instalarse antes de: Ninguno

M4

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make y Sed

El banco de pruebas depende de: Diffutils

Debe instalarse antes de: Autoconf y Bison

Man-DB

Para su instalación depende de: Bash, Berkeley DB, Binutils, Bzip2, Coreutils, Flex, GCC, Gettext, Glibc, Grep, Groff, Gzip, Less, Make y Sed

El banco de pruebas depende de: No se ejecuta. Requiere un paquete extra con el banco de pruebas de Man-DB.

Debe instalarse antes de: Ninguno

Make

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Gettext, Glibc, Grep, Make, Sed y Texinfo

El banco de pruebas depende de: Perl

Debe instalarse antes de: Ninguno

Mktemp

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Patch y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Module-Init-Tools

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Sed y Zlib

El banco de pruebas depende de: File, Findutils y Gawk

Debe instalarse antes de: Ninguno

Ncurses

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Gawk, GCC, Glibc, Grep, Make, Patch y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Bash, GRUB, Inetutils, Less, Procps, Psmisc, Readline, Texinfo, Util-Linux y Vim

Patch

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Perl

Para su instalación depende de: Bash, Berkeley DB, Binutils, Coreutils, Gawk, GCC, Glibc, Grep, Groff, Make y Sed

El banco de pruebas depende de: Iana-Etc y Procps

Debe instalarse antes de: Autoconf

Procps

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Make y Ncurses

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Psmisc

Para su instalación depende de: Bash, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Readline

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make, Ncurses, Patch, Sed y Texinfo

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Bash

Sed

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Sed y Texinfo

El banco de pruebas depende de: Diffutils y Gawk

Debe instalarse antes de: E2fsprogs, File, Libtool y Shadow

Shadow

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Make y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Sysklogd

Para su instalación depende de: Binutils, Coreutils, GCC, Glibc, Make y Patch

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Sysvinit

Para su instalación depende de: Binutils, Coreutils, GCC, Glibc, Make y Sed

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Tar

Para su instalación depende de: Bash, Binutils, Bison, Coreutils, GCC, Gettext, Glibc, Grep, Inetutils, Make, Patch, Sed y Texinfo

El banco de pruebas depende de: Diffutils, Findutils y Gawk

Debe instalarse antes de: Ninguno

Tcl

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make y Sed

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

Texinfo

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch y Sed

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

Udev

Para su instalación depende de: Binutils, Coreutils, GCC, Glibc y Make

El banco de pruebas depende de: Findutils, Perl y Sed

Debe instalarse antes de: Ninguno

Util-Linux

Para su instalación depende de: Bash, Binutils, Coreutils, E2fprogs, GCC, Gettext, Glibc, Grep, Make, Ncurses, Patch, Sed y Zlib

El banco de pruebas depende de: No incluye un banco de pruebas.

Debe instalarse antes de: Ninguno

Vim

Para su instalación depende de: Bash, Binutils, Coreutils, Diffutils, GCC, Glibc, Grep, Make, Ncurses y Sed

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: Ninguno

Zlib

Para su instalación depende de: Bash, Binutils, Coreutils, GCC, Glibc, Grep, Make y Sed

El banco de pruebas depende de: Ninguno

Debe instalarse antes de: File, Module-Init-Tools y Util-Linux

Índice

Paquetes

Programas

Librerías

Guiones

Otros