Linux From Scratch

Linux From Scratch

Versión 4.1

Gerard Beekmans

Copyright © 1999−2003 por Gerard Beekmans, sobre el texto original.

Copyright © 2002−2003 por Proyecto LFS−ES, sobre la traducción al castellano.

Traducido por el proyecto LFS−ES http://www.escomposlinux.org/lfs−es

Versión de la traducción: FINAL del 28 de abril de 2003

Este libro describe el proceso para la creación de un sistema Linux desde cero, usando solamente las fuentes del software necesario.

Copyright (c) 2002−2003, Proyecto LFS−ES

El presente texto se distribuye bajo la Licencia GNU de documentación libre (GFDL). Para todo aquello no especificado en dicha licencia es de aplicación las condiciones de uso del documento original en el que se basa esta traducción, y que se citan a continuación.

Copyright (c) 1999−2003, Gerard Beekmans

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions in any form must retain the above copyright notice, this list of conditions and the following disclaimer.
- Neither the name of "Linux From Scratch" nor the names of its contributors may be used to endorse or promote products derived from this material without specific prior written permission.
- Any material derived from Linux From Scratch must contain a reference to the "Linux From Scratch" project.

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

OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Dedicatoria

Este libro está dedicado a la querida esposa de Gerard Beekmans, *Beverly Beekmans*.

Índice

Prólogo
- *Prefacio*
- *Quién puede querer leer este libro*
- *A quién puede que no le interese leer el libro*
- *Prerrequisitos*
- *Organización*
  - *Parte I – Introducción*
  - *Parte II – Preparativos para la construcción*
  - *Parte III – Construcción del sistema LFS*
  - *Parte IV – Apéndice*

I. Parte I – Introducción
- *1. Introducción*
  - *Agradecimientos*
  - *Lo que vamos a hacer*
  - *Convenciones utilizadas en este libro*
  - *Versión del libro*
  - *Servidores alternativos*
  - *Historial de modificaciones*
  - *Listas de correo y archivos*
  - *Servidores de noticias*
  - *FAQ*
  - *Información de contacto*
- *2. Información importante*
  - *Sobre $LFS*
  - *Sobre los SBUs*
  - *Cómo buscar ayuda*

II. Parte II – Preparativos para la construcción
- *3. Preparación de una nueva partición*
  - *Introducción*
  - *Crear una nueva partición*
  - *Crear un sistema de ficheros en la nueva partición*
  - *Montar la nueva partición*
- *4. Paquetes que hay que descargar*
  - *Introducción*
  - *Paquetes que hay que descargar*
- *5. Preparación del sistema LFS*
  - *Introducción*
  - *¿Por qué usamos enlazado estático?*
  - *Creación del directorio $LFS/static*
  - *Añadir el usuario lfs*
  - *Configuración del entorno*
  - *Instalación de Bash-2.05a*
  - *Instalación de Binutils-2.13.2*
  - *Instalación de Bzip2-1.0.2*
  - *Instalación de Diffutils-2.8.1*

Instalación de Fileutils−4.1
Instalación de Findutils−4.1
Instalación de Gawk−3.1.1
Instalación de GCC−3.2.1
Instalación de Grep−2.5
Instalación de Gzip−1.2.4a
Instalación de Make−3.80
Instalación de Patch−2.5.4
Instalación de Sed−4.0.5
Instalación de Sh−utils−2.0
Instalación de Tar−1.13
Instalación de Texinfo−4.3
Instalación de Textutils−2.1
Instalación de Util−linux−2.11y
III. Parte III − Construcción del sistema LFS
6. Instalación de los programas del sistema base
Introducción
Sobre los símbolos de depuración
Entrando al entorno chroot
Cambio del propietario
Creación de los directorios
Montando el sistema de ficheros /proc
Creación del fichero mtab
Creación de los enlaces simbólicos bash y sh
Creación de los ficheros de contraseñas y grupos
Creación de los dispositivos (Makedev−1.7)
Instalación de las cabeceras de Linux−2.4.20
Instalación de Man−pages−1.54
Instalación de Glibc−2.3.1
Instalación de GCC−3.2.1
Instalación de Zlib−1.1.4
Instalación de Findutils−4.1
Instalación de Gawk−3.1.1
Instalación de Ncurses−5.3
Instalación de Vim−6.1
Instalación de M4−1.4
Instalación de Bison−1.875
Instalación de Less−378
Instalación de Groff−1.18.1
Instalación de Textutils−2.1
Instalación de Sed−4.0.5
Instalación de Flex−2.5.4a
Instalación de Binutils−2.13.2
Instalación de Fileutils−4.1
Instalación de Sh−utils−2.0
Instalación de Gettext−0.11.5
Instalación de Net−tools−1.60
Instalación de Perl−5.8.0
Instalación de Texinfo−4.3
Instalación de Autoconf−2.57
Instalación de Automake−1.7.2

Instalación de Bash−2.05a
Instalación de File−3.39
Instalación de Libtool−1.4.3
Instalación de Bin86−0.16.3
Instalación de Bzip2−1.0.2
Instalación de Ed−0.2
Instalación de Kbd−1.08
Instalación de Diffutils−2.8.1
Instalación de E2fsprogs−1.32
Instalación de Grep−2.5
Instalación de Gzip−1.2.4a
Instalación de Man−1.5k
Instalación de Lilo−22.2
Instalación de Make−3.80
Instalación de Modutils−2.4.22
Instalación de Netkit−base−0.17
Instalación de Patch−2.5.4
Instalación de Procinfo−18
Instalación de Procps−3.1.5
Instalación de Psmisc−21.2
Instalación de Shadow−4.0.3
Instalación de Sysklogd−1.4.1
Instalación de Sysvinit−2.84
Instalación de Tar−1.13
Instalación de Util−linux−2.11y
Instalación de las páginas de manual de Linux−2.4.20
Instalación de Glibc−2.3.1
Comando chroot revisado
Instalación de LFS−Bootscripts−1.11
Configuración de los componentes del sistema
7. Preparando los guiones de arranque
Introducción
¿Cómo hacen estos guiones que funcione el proceso de arranque?
Configuración del guión setclock
¿Necesito el guión loadkeys?
Configuración del guión sysklogd
Configuración del guión localnet
Creación del fichero /etc/hosts
Configuración del guión network
8. Hacer el sistema LFS arrancable
Introducción
Creación del fichero /etc/fstab
Instalación de Linux−2.4.20
Hacer el sistema LFS arrancable
9. El final
El final
Registrarse
Arranque del sistema
Y ahora, ¿qué?
IV. Parte IV − Apéndices
A. Descripción de paquetes y dependencias

Introducción
Autoconf
Automake
Bash
Bin86
Binutils
Bison
Bzip2
Diffutils
E2fsprogs
Ed
File
Fileutils
Findutils
Flex
Gawk
GCC
Gettext
Glibc
Grep
Groff
Gzip
Kbd
Less
LFS−Bootscripts
Libtool
Lilo
Linux (el núcleo)
M4
Make
MAKEDEV
Man
Man−pages
Modutils
Ncurses
Netkit−base
Net−tools
Patch
Perl
Procinfo
Procps
Psmisc
Sed
Shadow
Sh−utils
Sysklogd
Sysvinit
Tar
Texinfo
Textutils
Util−linux

Vim
Zlib

Prólogo

Prefacio

Después de haber usado diferentes distribuciones de Linux, nunca estuve satisfecho con ninguna de ellas. No me gustaba la forma en la que estaban organizados los guiones de arranque, o no me gustaba la configuración por defecto de ciertos programas, y cosas por el estilo. Llegué a darme cuenta de que si quería estar completamente satisfecho con algún sistema Linux, tenía que construir el mío propio desde cero, usando, idealmente, sólo el código fuente. Sin utilizar paquetes precompilados de ninguna clase. Sin la ayuda de un CD−ROM o disco de arranque que instalase utilidades básicas. Utilizaría mi sistema Linux actual para construir el mío por mi cuenta.

Esta, en su momento, idea descabellada se presentó muy difícil y algunas veces casi imposible. Después de sortear toda clase de problemas de dependencias, de compilación, etc., creé un sistema Linux hecho a medida y completamente funcional. Llamé a este sistema LFS, que significa Linux From Scratch (Linux Desde Cero).

¡Espero que paséis buenos momentos trabajando en vuestro LFS!

––
Gerard Beekmans
gerard@linuxfromscratch.org

Quién puede querer leer este libro

Existen muchas razones por las que alguien podría querer leer este libro. La principal razón es instalar un sistema LFS. La pregunta que mucha gente podría hacer es "¿Por qué pasar por todo el embrollo de instalar manualmente un sistema desde cero cuando te puedes limitar a descargar una distribución ya existente?". Es una buena pregunta.

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. Y quizás lo más importante, cómo adaptarlo 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 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. Sólo están ahí gastando (precioso) espacio de disco. No es muy difícil conseguir un sistema LFS instalado en menos de 100 MB. ¿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; el espacio total de disco usado fue, aproximadamente, 8 MB. Con un repaso adicional para reducirlo, se podría llegar a 5 MB o menos. Intenta eso con una distribución normal.

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. Ahora tienes la oportunidad de cocinarla a tu gusto: asarla, cocerla, freírla, hacerla a la barbacoa, o comerla cruda.

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 como quieras.

Otra 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 tape ese agujero de seguridad. Hasta que examines el nuevo parche y lo construyas no tienes garantía de que ese nuevo paquete se haya construido correctamente y realmente solucione el problema (de forma adecuada). Nunca podrás saber realmente si un agujero de seguridad está solucionado a no ser que lo hagas por ti mismo.

A quién puede que no le interese leer el libro

Si no deseas construir tu propio sistema Linux desde cero probablemente no quieras leer este libro. Nuestra meta es construir los fundamentos de un sistema completo y usable. 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 Del Bash CÓMO)". 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 .

Mientras decides lo que vas a leer, considera tu objetivo. Si deseas construir un sistema Linux mientras aprendes un poco en el camino, entonces este libro es tu mejor elección. Si tu objetivo es estrictamente educacional, y no tienes planes para tu sistema terminado, entonces el "De La Puesta En Marcha Al Indicador Del Bash CÓMO" es, probablemente, mejor elección.

Podrás encontrar el "De La Puesta En Marcha Al Indicador Del Bash CÓMO" en http://personal.telefonica.terra.es/web/aus/linux/p2b/power2bash.html y el original "From Power Up To Bash Prompt HOWTO" en http://axiom.anu.edu.au/~okeefe/p2b/.

Prerrequisitos

Este libro asume que sus lectores tienen un buen conocimiento sobre la utilización e instalación de software en Linux. Antes de que empieces a construir tu sistema LFS, deberías leer los siguientes CÓMOs:

- Software−Building−HOWTO (Construcción de Software CÓMO)

  Esta es uns guía asequible sobre cómo construir e instalar las distribuciones de software UNIX "genéricas" bajo Linux. Este CÓMO está disponible en http://www.tldp.org/HOWTO/Software−Building−HOWTO.html.
- The Linux Users' Guide (La Guía del Usuario de Linux)

  Estas guía cubre el uso de un surtido de software Linux. Está disponible en castellano en http://es.tldp.org/Manuales−LuCAS/GLUP/glup_0.6−1.1−html−1.1 y el original en inglés se encuentra en http://espc22.murdoch.edu.au/~stewart/guide/guide.html.

Organización

Este libro se divide en las siguientes cuatro partes:

Parte I – Introducción

En la Parte I se explican algunas cosas importantes sobre cómo proceder durante la instalación, y facilita información sobre el propio libro (versión, historial de modificaciones, reconocimientos, listas de correso asociadas y más cosas).

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 guia a través de la construcción del sistema LFS: compilar e instalar todos los paquete uno por uno, activar los guiones de arranque e instalar el núcleo. El sistema básico Linux obtenido será los cimientos sobre los que podrás construir más software, ampliando tu sistema del modo que prefieras.

Parte IV – Apéndice

La Parte IV consiste sólo, por el momento, en un simple apéndice: un listado alfabético de todos los paquetes instalados, mostrando para cada uno su localización oficial de descarga, su contenido y sus dependencias de instalación.

(Gran parte del Apéndice A está integrado en las partes II y III. Esto agranda algo el libro, pero creemos que facilta su lectura. De esta forma no tienes que dirigirte al apéndice mientras haces la instalación. Este ir y venir puede ser un fastido, sobre todo si estás leyendo la versión del libro basada en texto.)

I. Parte I – Introducción

Índice

1. Introducción
2. Información importante

Capítulo 1. Introducción

Agradecimientos

Agradecemos 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 CVS, listas de correo y espacio web, que tan vitales son para realizar nuestro trabajo.
- A todos aquellos que leen nuestros textos con interes, pues es para ellos para quien los escribimos.

Agradecemos a las siguientes personas y organizaciones su contribución al proyecto LFS:

- Mark Stone <mstone@linux.com> por donar el servidor linuxfromscratch.org.
- VA Linux Systems por proveer espacio de rack y ancho de banda al servidor linuxfromscratch.org.
- Fredrik Danerklint por mantener el servidor alternativo se.linuxfromscratch.org.
- Tim Jackson <tim@idge.net> por mantener el servidor alternativo linuxfromscratch.idge.net.
- Hagen Herrschaft <hrx@hrxnet.de> por mantener los servidores alternativos de.linuxfromscratch.org y por donar un sistema P4−2.2Ghz al proyecto LFS.
- UK Mirror Service por mantener el servidor alternativo linuxfromscratch.mirror.ac.uk.
- Guido Passet <guido@primerelay.net> por mantener los servidores alternativos www.nl.linuxfromscratch.org y ftp.snt.utwente.nl.
- Timothy Bauscher <timothy@linuxfromscratch.org> por ser de gran ayuda en la edición de este libro.
- Mark Hymers <markh@linuxfromscratch.org> por su enorme ayuda en la edición de este libro.
- Marc Heerdink <marc_heerdink@softhome.net> también por su gran ayuda en la edición de este libro.
- DREAMWVR.COM por su continuado respaldo donando varios recursos al proyecto LFS y a subproyectos relacionados.
- Jan Niemann <jan.niemann@tu.bs.de> por mantener el servidor alternativo www.de.linuxfromscratch.org.
- Torsten Westermann <westermann@linux−provider.net> por mantener el servidor alternativo lfs.linux−provider.net.
- Ian Chilton <ian@ichilton.co.uk> por mantener los servidores alternativos www.us.linuxfromscratch.org y www.linuxfromscratch.co.uk.
- Dag Stenstad <dag@stenstad.net> por proveer el servidor alternativo www.no.linuxfromscratch.org, y Ian Chilton <ian@ichilton.co.uk> por mantenerlo.
- Antonin Sprinzl <Antonin.Sprinzl@tuwien.ac.at> por mantener el servidor alternativo www.at.linuxfromscratch.org.
- Jason Andrade <jason@dstc.edu.au> por mantener el servidor alternativo www.au.linuxfromscratch.org.
- Ian Cooper <ian@wpi.edu> por mantener el servidor alternativo www.us2.linuxfromscratch.org.
- VA Linux Systems que, en nombre de Linux.com, donó al proyecto una estación de trabajo VA Linux 420 (anteriormente StartX SP2).
- Johan Lenglet <johan@linuxfromscratch.org> por gestionar el proyecto de traducción de LFS al francés.
- Jesse Tie−Ten−Quee <highos@linuxfromscratch.org> por donar una grabadora de CD Yamaha CDRW 8824E.
- O'Reilly por donar libros sobre SQL y PHP.
- Robert Briggs por donar los nombres de dominio linuxfromscratch.org y linuxfromscratch.com.

- Frank Skettino <bkenoah@oswd.org> de OSWD por crear el diseño inicial del sitio web LFS.
- Garrett LeSage <garrett@linux.com> por crear el logotipo de LFS.
- Dean Benson <dean@vipersoft.co.uk> por su ayuda financiera al establecer la organización sin ánimo de lucro LFS.
- Innumerables otros en varias listas de correo LFS que están haciendo posible este libro aportando sugerencias, leyendo el libro e informando de los errores.

Lo que vamos a hacer

Vamos a construir el sistema LFS utilizando una distribución ya instalada, como Debian, SuSE, Slackware, Mandrake, RedHat, etc. Utilizaremos un sistema Linux existente como plataforma de desarrollo porque necesitamos un compilador, enlazador (linker), editor de texto y otras herramientas para construir nuestro sistema. Por regla general, las herramientas que vamos a necesitar se instalan por defecto si durante la instalación de nuestra distribución Linux seleccionamos la opción de "desarrollo".

En cuando hayas descargado los paquetes que componen un sistema LFS, crearemos una nueva partición Linux nativa sobre la que se instalará el sistema LFS.

En el siguiente paso, Capítulo 5, discutiremos la instalación de una serie de paquetes que constituyen un juego básico de herramientas de desarrollo, que se usarán para instalar el sistema propiamente dicho, y que también son necesarios para resolver dependencias circulares (por ejemplo, necesitas un compilador para instalar un compilador y necesitas un intérprete de comandos para instalar un intérprete de comandos). En este capítulo los paquetes serán enlazados estáticamente.

El enlazado estático define un método de compilación que hace innecesaria la presencia de librerías tras la construcción del software. El programa resultante es capaz de funcionar por si mismo porque las porciones de código de las librerías que necesita el programa se meten dentro de él. Normalmente el software se enlaza dinámicamente. De esta manera se conserva más espacio en el disco y se mejora la eficiencia de varios programas. Nosotros enlazamos estáticamente los programas en el Capítulo 5 porque en teoría estaremos moviendo nuestro sistema de desarrollo a un entorno virtual donde no existirán las librerías mencionadas anteriormente. Si los programas se enlazan dinámicamente, nuestro entorno de desarrollo no funcionará. Ya que las librerías de las que estamos hablando están en nuestra distribución Linux, el objetivo del Capítulo 5 es construir un entorno de desarrollo en el que esas librerías no sean necesarias y que, por tanto, sea independiente de la distribución.

En el Capítulo 6 construiremos e instalaremos nuestro sistema final. Usaremos el programa chroot para entrar en un entorno virtual y ejecutar un nuevo intérprete de comandos cuyo directorio raíz será la partición donde construimos todos los programas del Capítulo 5. Esto es equivalente a reiniciar el sistema haciendo que el núcleo monte nuestra partición LFS como partición raíz. La razón de que utilicemos chroot en lugar de reiniciar es que crear un sistema estático arrancable requiere un trabajo adicional que no es necesario. Así, podemos seguir usando nuestro sistema anfitrión mientras construimos LFS. Mientras se está instalando el software, puedes simplemente cambiar a otra VC (consola virtual) o escritorio X y continuar usando tu ordenador como lo harías normalmente.

Cuando esté instalado todo el software del Capítulo 6, los Capítulos 7, 8 y 9 nos ayudarán a terminar la instalación. En el Capítulo 7 configuraremos los guiones de arranque. En el Capítulo 8 construiremos nuestro núcleo (kernel) final y configuraremos el gestor de arranque. Y en el Capítulo 9 hay algunas sugerencias sobre lo que puedes hacer cuando acabes el libro. En ese momento puedes por fin reiniciar tu nuevo sistema LFS, y empezar a usarlo realmente.

En pocas palabras, este es el proceso. Encontrarás información detallada sobre los pasos que has de seguir en cada capítulo, a medida que avances. Si ahora hay algo que no veas muy claro, no te preocupes. Debería aclararse perfectamente un poco más adelante.

Por favor, lee con atención el Capítulo 2, ya que explica algunas cosas importantes que debes saber antes de comenzar a trabajar en el Capítulo 5 y posteriores.

Convenciones utilizadas en este libro

Para facilitar la comprensión se utilizan ciertas convenciones a lo largo del libro. Aquí hay unos ejemplos:

`./configure --prefix=/usr`

El texto con este estilo debe teclearse exactamente como aparece, a menos que se indique lo contrario. 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'`

El estilo de este texto (ancho fijo) representa salida por pantalla, probablemente como resultado de la ejecución de comandos; y también se usa para especificar nombres de archivo, como, por ejemplo `/etc/lilo.conf`.

Énfasis

Este tipo de texto se utiliza con varios fines en el libro, principalmente para poner de relieve puntos importantes y para dar ejemplos de qué se debe teclear.

http://www.linuxfromscratch.org/

Este tipo de texto se usa para hipervínculos, tanto al propio libro como a páginas externas (tales como direcciones de descarga, CÓMOs, sitios web, etc).

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

Este tipo de secciones se usa principalmente al crear archivos de configuración. El primer comando (en negrita) solicita al sistema que cree el archivo $LFS/etc/group a partir de lo que se teclee en las líneas siguientes, hasta encontrar la secuencia EOF. Por lo tanto, generalmente la sección entera debe teclearse tal cual.

Versión del libro

Esta es la traducción al castellano del libro LFS−BOOK versión 4.1 con fecha 28 de abril de 2003. Si la versión tiene más de un mes de antigüedad, es probable que haya una versión más nueva disponible para su descarga. Puedes buscar nuevas versiones en los sitios de descarga alternativos (mirrors) relacionados a continuación.

Servidores alternativos

El proyecto LFS tiene por todo el mundo un número de servidores alternativos para facilitar el acceso y hacer más conveniente para ti el acceso a las páginas web y la descarga de los paquetes requeridos. Por favor, visita el sitio web http://www.linuxfromscratch.org para consultar la lísta 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] − http://www.escomposlinux.org/lfs−es
- Cervera, España [126 Kbits] − http://www.macana−es.com

Historial de modificaciones

4.1 − 28 de abril de 2003

- Actualizado a:
  - autoconf−2.57
  - automake−1.7.2
  - bison−1.875
  - e2fsprogs−1.32
  - gawk−3.1.1−3.patch
  - gcc−3.2.1
  - glibc−2.3.1
  - groff−1.18.1
  - kbd−1.08
  - less−378
  - lfs−bootscripts−1.11
  - libtool−1.4.3
  - linux−2.4.20
  - make−3.80
  - man−1.5k−2.patch
  - man−pages−1.54
  - modutils−2.4.22
  - ncurses−5.3
  - procps−3.1.5
  - psmisc−21.2
  - sed−4.0.5
  - texinfo−4.3
  - util−linux−2.11y
- Añadido:

    - findutils−4.1−segfault.patch
    - glibc−2.3.1−libnss.patch
    - glibc−2.3.1−root−perl.patch
    - kbd−1.08.patch
    - man−1.5k−80cols.patch
    - man−1.5k−manpath.patch
    - man−1.5k−pager.patch
- Eliminado:

    - gcc−3.2.1−nofixincludes−2.patch
    - glibc−2.3.1.patch
    - kbd−1.06−3.patch
    - man−1.5k−2.patch
    - ncurses−5.2−2.patch
- 3 de febrero de 2003 [gerard]: Actualizado a lfs−bootscripts−1.11 para implementar los cambios a mtab de Seth Klein's (ver aquí abajo).
- 3 de febrero de 2003 [sklein]: Capítulo 06: Cambiado el enlace simbólico `/etc/mtab` por un fichero.
- 29 de enero de 2003 [gerard]: Capítulo 06 − GCC: Cambiada la opción de instalación a install−no−fixedincludes.
- 29 de enero de 2003 [gerard]: Revertido a binutils−2.13.2 debido a un error de GCC por el que no puede procesar la cadena 2.13.2.1 de la versión de binutils. El único cambio entra las versiones 2.13.2 y 2.13.2.1 se centra en la documentación, por lo en realidad no se revierte el código. Más adelante se añadirá al libro una mejor solución.
- 27 de enero de 2003 [gerard, timothy, billy]: Convertidas las páginas de instalación de software al nuevo formato. Se a añadido también a dichas páginas las instrucciones de configuración procedentes de "Configuración del software esencial".
- 22 de enero de 2003 [timothy]: Capítulo 06 − Configuración de los programas esenciales: Corregida la ruta a las fuentes del núcleo para que las encuentre keymap.
- 10 de enero de 2003 [gerard]: Añadido un nuevo comando chroot antes de la segunda compilación de Glibc que debe usarse desde este punto en adelante.
- 9 de enero de 2003 [timothy]: Apéndice A − Gzip: Añadida la URL del parche.
- 9 de enero de 2003 [timothy]: Capítulo 05 − Findutils: Eliminado el modificador −D_GNU_SOURCE por ahora debido a que rompe la compilación; cuatro personas han informado de este fallo.
- 8 de enero de 2003 [timothy]: Capítulo 05 − Findutils: Añadido un / olvidado después de la línea de `configure`.
- 6 de enero de 2003 [gerard]: Capítulo 06 − Bison: eliminada la creación del guión yacc. Bison lo instala ahora por defecto.
- 6 de enero de 2003 [gerard]: Actualizado a Binutils−2.13.2.1, Bison−1.875 y Man−pages−1.54
- 6 de enero de 2003 [gerard]: Capítulos 05+06 − Findutils: Añadido *CPPFLAGS=−D_GNU_SOURCE* para su correcta compilación en arquitecturas que no sean la x86.
- 6 de enero de 2003 [gerard]: Capítulo 06 − Zlib: Añadida la variable CFLAGS para definir −fPIC, así las librerías dinámicas podrán conpilarse correctamente en todo momento.
- 5 de enero de 2003 [timothy]: Capítulo 05 − Aplicada una versión revisada del parche de Alex para dividir la página *Instalar todo el software como usuario sin privilegios* en dos páginas: *Añadir el usuario lfs* y *Establecer el entorno*.
- 2 de enero de 2003 [gerard]: Capítulo 05 − Todos los paquetes cuyo guión configure soporte ahora la variable de entorno LDFLAGS la usamos, en vez de pasarle la variable a **`make`**.
- 2 de enero de 2003 [gerard]: Capítulo 06 − Corregido el parche de gawk. **`make uninstall`** ya no vaciará el directorio `/usr/bin`. Al mismo tiempo, se renombra `/usr/share/gawk` a `/usr/share/gawk−3.1.1`.

- 2 de enero de 2003 [gerard]: Sustituido el mega−parche de glibc−2.3.1 por dos parches separtados (glibc−2.3.1−root−perl.patch y glibc−2.3.1−libnss.patch).
- 2 de enero de 2003 [gerard]: Sustituido el mega−parche de man−1.5k por tres parches separados (man−1.5k−80cols.patch, man−1.5k−manpath.patch y man−1.5k−pager.patch).
- 1 de enero de 2003 [gerard]: Capítulo 06 – Glibc Segunda Fase: Corregido un error tipográfico en la instalación de las páginas de manual de linuxthreads.
- 1 de enero de 2003 [gerard]: Capítulo 06 – Núcleo Linux: Las páginas de manual no se instalan aquí porque necesitan Perl. Movido al final del capítulo 6.
- 31 de diciembre de 2002 [gerard]: Capítulo 06 – Man: Actualizado el parche para que las páginas de manual se formateen correctamente en patallas con más de 80 columnas.
- 31 de diciembre de 2002 [gerard]: Capítulo 06 – Linux: Añadido *make mandocs* para crear las páginas de manual y copiarlas a `/usr/share/man/man9`
- 31 de diciembre de 2002 [gerard]: Apéndice A – Bzip2: Cambiada la localización de descarga a http://sources.redhat.com/bzip2
- 31 de diciembre 2002 [gerard]: Capítulo 06: Añadida la segunda instalación de Glibc al final del capítulo. Eliminada la instalación aparte de las páginas de manual de linuxthreads y movida a la segunda instalación de Glibc.
- 31 de diciembre de 2002 [gerard]: Actualizado a Glibc−2.3.1
- 31 de diciembre de 2002 [gerard]: Capítulo 05 – GCC: Eliminado el parche nofixincludes y usada la opción incorporada a make *install−no−fixedincludes*.
- 31 de diciembre de 2002 [gerard]: Capítulo 05 – GCC: Eliminado *HAVE_GAS_HIDDEN*, añadidos los modificadores de configure *−−with−ld y −−with−as*.
- 29 de diciembre de 2002 [timothy]: Actualizado a binutils−2.13.2, procps−3.1.5.
- 29 de diciembre de 2002 [timothy]: Capítulo 05: Cambiados todos los LDFLAGS=−static por LDFLAGS="−static".
- 29 de diciembre de 2002 [timothy]: Capítulo 06 – Flex: Añadidos enlaces simbólicos de libfl.a a libl.a.
- 20 de diciembre de 2002 [timothy]: Actualizado a sed−4.0.5.
- 18 de diciembre de 2002 [timothy]: Actualizado a procps−3.1.4.
- 17 de diciembre de 2002 [timothy]: Capítulos 5 y 6: Modificado el párrafo sobre desempaquetar los parches pues ya no están comprimidos.
- 15 de diciembre de 2002 [timothy]: Actualizado a autoconf−2.57, automake−1.7.2, binutils−2.13.1, e2fsprogs−1.32, gcc−3.2.1, libtool−1.4.3, linux−2.4.20, modutils−2.4.22, procps−3.1.3, sed−4.0.4, texinfo−4.3, util−linux−2.11y.
- 15 de diciembre de 2002 [timothy]: Capítulo 06 – Glibc: Eliminado el aviso sobre −−enable−kernel.
- 10 de diciembre de 2002 [gerard]: Capítulo 04 – Cambiados todos los enlaces por enlaces al proyecto Freshmeat.net project links, eliminado el paquete lfs−pacakges tarball. Esto se ha hecho debido a que el FTP de LFS FTP ya no contiene los paquetes, en su lugar hay que ir al lugar de descarga de los paquetes para conseguirlos.
- 5 de diciembre de 2002 [gerard]: Capítulo 08 – Renombrado *usbdevfs* a *usbfs* pues los chicos del núcleo han hecho este cambio para evitar confusiones con *devfs*.
- 3 de diciembre de 2002 [gerard]: Capítulo 05 – Sed: Añadido −−disable−nls
- 3 de diciembre de 2002 [gerard]: Capítulo 03 – Creando los sistemas de ficheros: Añadida una nota de que debe ejecutarse **`mkswap`** si se crea una nueva partición de intercambio.
- 3 de diciembre de 2002 [gerard]: Capítulo 06 – Bzip2: Eliminadas las líneas innecesarias que primero crean un enlace simbólico y luego lo eliminan.
- 3 de diciembre de 2002 [gerard]: Apéndice A – Bzip2: Actualizada la URL de descarga.
- 3 de diciembre de 2002 [gerard]: Capítulo 06 – Groff: Eliminadas las variables *PROCESSEDEXAMPLEFILES=""*.
- 25 de octubre de 2002 [timothy]: Prólogo: Añadida la sección de "Prerrequisitos".
- 25 de octubre de 2002 [timothy]: Capítulo 09: Añadida la sección ¿Y ahora, qué?.

- 25 de octubre de 2002 [timothy]: Eliminado el Apéndice B.
- 25 de octubre de 2002 [timothy]: Capítulo 02: Eliminada la sección Qué Plataforma.
- 23 de octubre de 2002 [timothy]: Intercambiados el Capítulo 3 y el Capítulo 4.
- 23 de octubre de 2002 [timothy]: Capítulo 02: Eliminadas las seccciones "Dónde almacenar los programas descargados"" y "Cómo instalar los programas".
- 23 de octubre de 2002 [timothy]: Actualizado a bison−1.75, sed−4.0. Movido m4 antes de bison para cumplir su dependencia.
- 21 de octubre de 2002 [timothy]: Capítulo 06 − Linux−2.4.19: Sustituidos los comandos **`mkdir /usr/include/asm`** y **`cp`** con **`cp -HR`**.
- 21 de octubre de 2002 [timothy]: Añadido findutils−4.1−segfault.patch para corregir una violación de segmento en locate cuando encuentra una ruta muy larga.
- 21 de octubre de 2002 [timothy]: Añadido libtool−1.4.2.patch para corregir una incompatibilidad entre Autoconf 2.53 y Libtool 1.4.x.
- 21 de octubre de 2002 [timothy]: Actualizado a automake−1.7.1, modutils−2.4.21, man−pages−1.53, kbd−1.08, util−linux−2.11w, autoconf−2.54, e2fsprogs−1.29, groff−1.18.1, psmisc−21.2, less−378, procps−3.0.4, make−3.80, ncurses−5.3.
- 20 de octubre de 2002 [timothy]: Descomprimidos los parches.
- 13 de octubre de 2002 [markh]: Capítulo 05 − Bzip2: Añadido −s a los argumentos de CC por consistencia.
- 6 de octubre de 2002 [timothy]: Cambiado gcc−core y gcc−g++.
- 6 de octubre de 2002 [timothy]: Capítulo 06 − Aplicado un parche con correciones gramaticales de Bill Maltby.

4.0 − 5 de octubre de 2002

- 3 de octubre de 2002 [gerard]: Capítulo 06 − Linuxthreads: En lugar de **`cd man`** usar la opción *−C* para **`make`** que tiene más sentido (y también es más corto).
- 29 de septiembre de 2002 [gerard]: Capítulo 05 − GCC: Corregido el antiguo parche nofixincludes y reinstalado.
- 29 de septiembre de 2002 [markh]: Capítulo 05 − Bash: Corregida la referencia a Debian que trataba sobre la antigua versión estable (potato) y no puede aplicarse a la actual (woody). Notificado por h2k1 en #lfs.

4.0−RC1 − 28 de septiembre de 2002

- Actualizado a:
  - automake−1.6.3
  - bin86−0.16.3
  - binutils−2.13
  - bison−1.35
  - diffutils−2.8.1
  - file−3.39
  - gawk−3.1.1
  - gcc−3.2
  - gettext−0.11.5
  - groff−1.18
  - gzip−1.2.4b.patch
  - lfs−bootscripts−1.10
  - linux−2.4.19
  - MAKEDEV−1.7

    - man−1.5k
    - man−pages−1.52
    - modutils−2.4.19
    - ncurses−5.2−2.patch
    - perl−5.8.0
    - psmisc−21
    - texinfo−4.2
    - textutils−2.1
    - util−linux−2.11u
- Añadido:

    - ed−0.2.patch
    - fileutils−4.1.patch
    - gawk−3.1.2.patch
    - gcc−3.2.patch
    - gcc−3.2−nofixincludes.patch
    - glibc−2.2.5−2.patch
    - gzip−1.2.4b.patch
    - kbd−1.06−3.patch
    - man−1.5k.patch
    - ncurses−5.2.patch
    - procps−2.0.7.patch
    - sh−utils−2.0−hostname.patch
    - vim−6.1.patch
    - zlib−1.1.4
- Eliminado:

    - gzip−1.2.4a.patch
    - kbd−1.06−2.patch
    - reiserfsprogs−3.x.1b
- 28 de septiembre de 2002 [gerard]: Capítulo 05 − GCC: Añadido el parche nofixincludes para evitar que el guión se ejecute en el capítulo 05. Debe ejecutarse en el Capítulo 06, por lo que lo necesitamos en un parche aparte.
- 28 de septiembre de 2002 [gerard]: Capítulo 06 − Man: Sustituida la expresión sed por un parche normal.
- 28 de septiembre de 2002 [gerard]: Capítulo 06 − Bzip2: Eliminado *PREFIX=/usr* del comando **`make install`** porque *PREFIX* está establecido realmente a */usr* por defecto.
- 28 de septiembre de, 2002 [gerard]: Capítulo 06 − Vim: Eliminada la nota sobre el cumplimiento del FHS. Es erronea porque Vim no utiliza para nada el localstatedir.
- 28 de septiembre de 2002 [timothy]: Aplicado un parche gramatical de Bill Maltby. Cambiado "$LFS" a "LFS" cuando se habla de la variable de entorno LFS.
- 23 de septiembre de 2002 [timothy]: Aplicados unos parches con correcciones gramaticales de Bill Maltby.
- 23 de septiembre de 2002 [timothy]: Añadido **−** antes de las opciones de **`tar`** (por claridad).
- 22 de septiembre de 2002 [timothy]: Capítulo 06: Aplicado un parche con correcciones gramaticales de Alex.
- 21 de septiembre de 2002 [timothy]: Capítulo 02: Aplicado un parche con correcciones gramaticales de Bill Maltby.
- 21 de septiembre de 2002 [timothy]: Capitulo 06 − Zlib: **`mv`** las librerías compartidas a /lib.
- 20 de septiembre de 2002 [timothy]: Capítulo 05 − GCC: Eliminada la opción **`--enable-threads=posix`** pues no construimos un compilador C++ en este capítulo.

- 18 de septiembre de 2002 [timothy]: Capítulo 05 – Introducción: Eliminado un párrafo sobre enlazado estático pues parece confuso y está cubierto mejor y con más detalle en *¿Por qué Estático?*.
- 18 de septiembre de 2002 [timothy]: Capítulo 08 – Linux: Eliminado el comando **cd**.
- 18 de septiembre de 2002 [timothy]: Capítulo 06 – Ncurses: Eliminada la explicación del antiguo comando **mv /lib/*.a /usr/lib**.
- 13 de septiembre de 2002 [gerard]: Capítulo 06 – Shadow: Añadido −−libdir=/usr/lib a las opciones del guión configure. De este modo se genera una `libshadow.la` correcta. Cambiado también el comando **mv** para mover todos los ficheros `libshadow.so*` al directorio `/lib`. Los ficheros `lib*a` todavía están en el directorio `/usr/lib`.
- 13 de septiembre de 2002 [gerard]: Capítulo 06 – Man: Añadida otra expresión al comando **sed** que modifica el fichero `man.conf`. La expresión añadida comenta la línea *MANPATH /usr/man* que provoca resultados duplicados cuando se utiliza el comando **whatis**.
- 13 de septiembre de 2002 [gerard]: Capítulo 06: Añadida la instalación de las *Páginas de manual de Linux Threads* después de la instalación de Perl.
- 12 de septiembre de 2002 [gerard]: Capítulo 06 – Crear el enlace mtab: Hacer el comando **ln** como **ln −sf** para que el fichero `/etc/mtab` existente, creado por el comando **mount**, sea eliminado antes de recrearlo como enlace simbólico.
- 12 de septiembre de 2002 [gerard]: Capítulo 06 – Sh−utils: Añadido el parche sh−utils−hostname que suprime la construcción del programa hostname. Se hace esto porque el programa hostname del paquete net−tools es superior a esta versión.
- 12 de septiembre de 2002 [gerard]: Capítulo 06 – Gawk: Actualizado el parche para Gawk. Ahora cambia también la localización del directorio *DDEFPATH*.
- 12 de septiembre de 2002 [gerard]: Capítulo 06 – Procps: Añadido un parche que corrige un problema con las locales que hacen que **top** falle con ciertos ajustes de locale.
- 12 de septiembre de 2002 [timothy]: Capítulo 04 – Creando un sistema de ficheros: Referencia a sistemas de ficheros alternativos en BLFS.
- 12 de septiembre de 2002 [gerard]: Eliminados todos los enlaces simbólicos `/usr/lib/*.so` superfluos de las instalaciones de las librerías.
- 12 de septiembre de 2002 [gerard]: Actualizado a lfs−bootscripts−1.10
- 12 de septiembre de 2002 [gerard]: Capítulo 06 – Configuración de Sysvinit: Cambiada la línea de sulogin a *once* en lugar de tenerlo recargando. De esta forma se comportará como se espera (por ejemplo, un CTRL+D continuará en lugar de reiniciar sulogin).
- 12 de septiembre de 2002 [gerard]: Capítulo 06 – GCC: Añadida la opción *−−enable−clocale=gnu* para asegurar que la librería C++ utiliza el modo locale correcto.
- 11 de septiembre de 2002 [timothy]: Prólogo: Cambios gramaticales.
- 8 de septiembre de 2002 [timothy]: Capítulo 06: Aplicado el parche con cambios gramaticales de Alex.
- 7 de septiembre de 2002 [timothy]: Capítulo 06 – Gzip: Añadido gzip−1.2.4b.patch.
- 7 de septiembre de 2002 [timothy]: Capítulo 05 – Textutils: Añadido re_max_failures2 para sistemas antiguos.
- 2 de septiembre de 2002 [timothy]: Capítulo 06 – Bash: Eliminada la creación del enlace simbólico sh. Creando los enlaces simbólicos bash y sh: Añadido el enlace simbólico /bin/bash, enlazado sh a bash. Gzip, Sysvinit, Util−Linux: Acortados los comandos cp. Makedev: Eliminada la creación y eliminación de /bin/bash. Man: Modificada la sentencia sed para editar la llamada a less, así SGR funcionará
- 1 de septiembre de 2002 [timothy]: Capítulo 06 – Sobre los símbolos de depuración: Eliminada la información sobre eliminar los símbolos de /static. Man: Añadida una sentencia sed para evitar que groff utilice la secuencia de escape SGR.
- 1 de septiembre de 2002 [timothy]: Capítulo 05 – Instalando todo el software como usuario sin privilegios: Añadido $CC='gcc −s' para omitir la compilación de los símbolos en los paquetes estáticos.

- 30 de agosto de 2002 [timothy]: Capítulo 06 – Makedev: Poner rm /bin/bash después de la creación de los dispositivos. Perl: Eliminada la información sobre el antiguo parche.
- 30 de agosto de 2002 [timothy]: Capítulo 05 – GCC: Reañadido HAVE_GAS_HIDDEN; eliminado −−enable−__cxa−atexit que era incorrecto e innecesario en este capítulo; añadida información sobre el parche.
- 26 de agosto de 2002 [gerard]: Añadido un nuevo parche a Glibc e introducido un parche para GCC.
- 26 de agosto de 2002 [gerard]: Actualizado a automake−1.6.3, gcc−3.2, groff−1.18, makedev−1.7, perl−5.8.0, util−linux−2.11u
- 22 de agosto de 2002 [timothy]: Apéndice: Añadidas las URL olvidadas de los parches.
- 18 de agosto de 2002 [timothy]: Capítulos 05 y 06: Cambiado ln −sf a ln −s donde es posible.
- 18 de agosto de 2002 [timothy]: Capítulo 06 – Binutils: cp libiberty.h después del install, pues es necesario para cierto software. Shadow: añadido un comando para eliminar el programa groups instalado por Shadow pues Sh−utils instala un (mejor) programa groups.
- 18 de agosto de 2002 [timothy]: Capítulo 05 – Sh−utils: Re−añadido sh−utils−2.0.patch.
- 16 de agosto de 2002 [markh]: Capítulo 06 – Poner man−pages justo después de la instalación de las cabeceras del núcleo.
- 15 de agosto de 2002 [markh]: Capítulo 06 – Poner la instalación de MAKEDEV antes de glibc y eliminar la creación temporal de `/dev/null`, pues ya no es necesario.
- 15 de agosto de 2002 [timothy]: Capítulo 04 – Preparando una nueva partición: mencionado que la partición de intercambio puede compartirse entre el LFS y el sistema anfitrión. Cambios gramaticales.
- 13 de agosto de 2002 [gerard]: Capítulo 06: Eliminada la opción *−−with−curses* en la instalación de Bash, pues es innecesaria aquí.
- 9 de agosto de 2002 [timothy]: Actualizado a modutils−2.4.19, linux−2.4.19, gettext−0.11.5, binutils−2.13, textutils−2.1.
- 9 de agosto de 2002 [timothy]: Capítulo 06 – Vim: cambiado el enlace a los editores alternativos de las recetas al BLFS.
- 8 de agosto de 2002 [gerard]: Capítulo 06 – Ncurses: eliminada la opción *−−disable−termcap* de configure. Ahora termcap está desactivado por defecto, así que no necesitamos esta opción (se quedó aquí de otros tiempos en que si era necesaria).
- 8 de agosto de 2002 [gerard]: Capítulo 06 – Linux: Añadido el comando **`cp include/asm−generic /usr/include`**. Hay programas que usan estos ficheros, al igual que las cabeceras del directorio `asm` puede que se dividan en el futuro y se coloquen en `asm−generic`.
- 8 de agosto de 2002 [gerard]: Appendix A – Gettext: añadida la descripción perdida del programa msgcat.
- 4 de agosto de 2002 [timothy]: Añadido zlib−1.1.4.
- 3 de agosto de 2002 [timothy]: Actualizado a man−pages−1.52, man−1.5k, gettext−0.11.4, modutils−2.4.18.
- 29 de julio de 2002 [timothy]: Eliminado Reiserfsprogs. Actualizado a util−linux−2.11t y file−3.39. Capítulo 04 & 05 – Creando una nueva partición, Introducción, Por qué estático: cambios gramaticales. Diffutils, Fileutils, Grep, Texinfo: poner LDFLAGS=−static antes de configure en lugar de como una argumento de make. GCC: añadido HAVE_GAS_HIDDEN a auto−host.h.
- 29 de julio de 2002 [timothy]: Capítulo 06 – Glibc: añadida la opción −−disable−profile.
- 29 de julio de 2002 [timothy] Capítulo 08 – Linux: añadida información sobre los módulos y la documentación del núcleo.
- 29 de julio de 2002 [timothy]: Capítulo 09 – Reiniciando el sistema: añadido un comando para eliminar el directorio static.
- 8 de julio de 2002 [timothy]: Capítulo 09 – Reiniciando el sistema: Señalar a BLFS como siguiente paso.
- 3 de julio de 2002 [timothy]: Capítulo 06 – Sysvinit: Simplificado el comando de sed y actualizada la descripción de la instalación debido a que ahora init muestra "Sending processes" en lugar de "Sending all processes".

- 2 de julio de 2002 [markh]: Cambio interno – Hacer que todos los parches usen una entidad &package−patch−version; y eliminar todas las referencias directas a la versión de los parches.
- 30 de junio de 2002 [timothy]: Actualizado a man−pages−1.51 y automake−1.6.2
- 24 de junio de 2002 [timothy]: Capítulo 06 – Shadow, Util−linux, LFS−Bootscripts: Actualizado el contenido de los paquetes.
- 23 de junio de 2002 [timothy]: Capítulos 05 y 06 – Net−tools, Perl, Texinfo, Autoconf, Automake, File, Libtool, Bin86, Vim, Linux, Bison, Less, Man−pages, Groff, Bzip2, E2fsprogs, Grep, Lilo, Modutils, Procps, Psmisc, Reiserfsprogs: Actualizado el contenido de los paquetes.
- 23 de Junio del 2002 [timothy]: M4, Bzip2,File, E2fsprogs: Añadido "última versión comprobada" para uniformidad. GCC: Eliminados los programas específicos para i686.
- 16 de junio de 2002 [timothy]: Capítulo 06 – Gettext: Actualizado el contenido del paquete.
- 14 de junio de 2002 [timothy]: Capítulos 05 y 06 – Binutils, Bzip2, Diffutils, Grep: Actualizado el contenido de los paquetes. GCC: Actualizada la descripción de c++filt.
- 13 de junio de 2002 [timothy]: Capítulo 09 – El Final: Cambiado $LFS/etc/lfs−4.1 por $LFS/etc/lfs y poner el número de la versión dentro de este fichero.
- 12 de junio de 2002 [timothy]: Capítulo 05 – GCC: Modificadas las instrucciones de construcción y las explicaciones de los comandos para construir sólo el compilador C. El compilador C++ no es necesario hasta la segunda construcción de GCC.
- 12 de junio de 2002 [timothy]: Capítulo 06 – Shadow: Cambios gramaticales.
- 11 de junio de 2002 [timothy]: Capítulos 05 y 06 – Gawk: Creada la lista de contenidos del paquete y las descripciones. Fileutils: Eliminado un párrafo confuso sobre el parche de fileutils. GCC: Actualizado el contenido del paquete.
- 11 de junio de 2002 [timothy] Todo el software: Actualizado el espacio requerido en disco.
- 9 de junio de 2002 [markh]: Capítulo 06 – Creando Directorios: Cambiado usr,usr/local a sólo usr/local, pues usamos la opción −p de mkdir que creará el directorio usr de todas formas.
- 7 de junio de 2002 [timothy] Capítulo 06 – Reiserfsprogs: añadida la descripción de unpack.
- 7 de junio de 2002 [timothy] Capítulo 02 – Cómo buscar ayuda: Mencionadas las FAQ.
- 6 de junio de 2002 [markh] – Capítulo 05 – Modificar explicaciones debido al cambio a /static.
- 5 de junio de 2002 [timothy] Prólogo – Quién puede no querer leer este libro: aplicada una versión revisada del parche de gramática de Scot.
- 5 de junio de 2002 [timothy] Capítulo 09 – Reiniciando el sistema, Lilo, Bootscripts: mencionar los autores de las recetas. Capítulo 06 – Vim: actualizada la URL de la receta. Capítulo 05 – Gawk: para evitar confusiones, mencionado el parche que se aplicará en el Capítulo 06.
- 3 de junio de 2002 [timothy] Capítulo 01 – FAQ: editada para incluir la notificación de errores tipográficos.
- 31 de mayo de 2002 [gerard] Capítulo 05 – Findutils: Añadido el arreglo de CPPFLAGS...re_max_failures necesario en sistemas Glibc−2.1.
- 30 de mayo de 2002 [markh]: Capítulos 05 y 06 – Actualizado a binutils−2.12.1.
- 30 de mayo de 2002 [markh]: Capítulo 05 – Bash: Eliminada la sección sobre "los últimos dos comandos se ejecutan igualmente" porque ya no usamos esos comandos a los que se refiere.
- 30 de mayo de 2002 [gerard]: Capítulo 06 – Glibc: Sustituidos los arreglos de sed por un parche normal.
- 30 de mayo de 2002 [gerard]: Capítulo 06 – Gawk: Sustituidos los arreglos de sed por un parche normal.
- 30 de mayo de 2002 [gerard]: Capítulo 05 – Fileutils: Sustituidos los arreglos de sed por un parche normal.
- 30 de mayo de 2002 [gerard]: Capítulo 06 – Ed: Sustituidos los arreglos de sed por un parche normal.
- 28 de mayo de 2002 [gerard]: Capítulo 06 – Cambiando el propietario: eliminado el comando explícito chown `/lost+found`.Esto se hace con el primer comando ahora que proc ya no se monta en el Capítulo 5.

- 27 de mayo de 2002 [gerard]: Actualizado a ncurses−5.2−2.patch (este parche es más pequeño que el usado antes).
- 26 de mayo de 2002 [gerard]: Actualizado a: automake−1.6.1, bin86−0.16.3, file−3.38, gawk−3.1.1, gcc−3.1, gettext−0.11.2, modutils−2.4.16, psmisc−21 y util−linux−2.11r. Añadidos unos parches para ncurses, perl y vim que arreglan su compilación con gcc−3.1.
- 26 de mayo de 2002 [gerard]: Capítulos 05 y 06 − Binutils: Eliminado el establecimiento del directorio de herramientas en el capítulo 05−binutils, movida su descripción al capítulo 06−binutils.
- 26 de mayo de 2002 [gerard]: Capítulo 05 − Gawk y Findutils: simplificada la instalación al eliminar las modificaciones de libexecdir. Podemos dejar que se cree `$LFS/static/libexecdir`. El directorio `$LFS/static` es temporal de todas formas, así que no importa el aspecto que tenga.
- 26 de mayo de 2002 [gerard]: Capítulo 06 − Creando directorios: eliminado el comando **`cd /`** y cambiados los dos comandos **`chmod`** para usar en su lugar rutas absolutas.
- 25 de mayo de 2002 [markh]: Capítulo 06 − Algunas correcciones menores relacionadas con la eliminación de la variable $LFS donde no es necesaria.
- 23 de mayo de 2002 [gerard]: Implementado el keep_chap5_and_chap6_sep lfs−hint. Mayores cambios: añadidos findutils y util−linux al capítulo 5, instalar todo lo del capítulo 5 en `$LFS/static` y reordenar la instalación de los paquetes en el capítulo 6 la inclusión en el código de rutas equivocadas (ficheros ubicados en `$LFS/static`).
- 23 de mayo de 2002 [gerard]: Apéndice A − E2fsprogs: Añadidas algunas descripciones.
- 23 de mayo de 2002 [gerard]: Apéndice A − Bin86: Añadidas algunas descripciones.
- 23 de mayo de 2002 [gerard]: Apéndice A − Flex: Añadidas algunas descripciones.
- 23 de mayo de 2002 [gerard]: Apéndice A − Glibc: Añadidas algunas descripciones.
- 18 de mayo de 2002 [gerard]: Apéndice A − E2fsprogs: Añadidas algunas descripciones.
- 18 de mayo de 2002 [gerard]: Apéndice A − Glibc: Añadidas algunas descripciones.
- 17 de mayo de 2002 [markh]: Cambiados todos los chown X.X a chown X:X que es menos propicio a crear problemas (según la documentación de chown).
- 16 de mayo de 2002 [gerard]: Capítulo 01 − Servidores alternativos: Añadido el acceso por http al servidor alternativo FTP idge.net
- 16 de mayo de 2002 [gerard]: Apéndice A − Glibc: Añadidas algunas descripciones.
- 15 de mayo de 2002 [markh]: Capítulo 05 − Bzip2. Cambiadas las instrucciones relacionadas con los enlaces duros en distribuciones antiguas de las instrucciones de gzip en el capítulo 05.
- 11 de mayo de 2002 [markh]: Varias modificaciones XML; básicamente alterar las etiquetas <ulink> para eliminar erróneos
 en la salida HTML.
- 9 de mayo de 2002 [gerard]: Apéndice A − Glibc: Poner descripciones desaparecidas.
- 6 de mayo de 2002 [gerard]: Capítulo 06 − Shadow: Fijada la localización del enlace simbólico `vigr` a `/usr/sbin`
- 2 de mayo de 2002 [gerard]: Capítulo 06 − Procps: Cambiadas las comillas simples por dobles (dos comillas simples pueden confundirse con una doble y provocar errores).
- 2 de mayo de 2002 [gerard]: Cambiado el comando **`cd dir && ln −sf`** a un único comando (como **`ln −sf bash $LFS/bin/sh`**) Lo mismo para las construcciones **`cd dir && mv/cp`** que ahora son sustituidas por **`mv $LFS/usr/bin/{bzcat,bzip2} $LFS/bin`**.
- 2 de mayo de 2002 [markh]: Eliminada la sección "Eliminando la antigua librería NSS".
- 1 de mayo de 2002 [gerard]: Eliminado todo lo relacionado con Glibc−2.0 − los parches para gzip y sh−utils patch y la copia de los ficheros libnss. También se cambian las construcciones **`export VAR=VALUE...unset VAR`** por **`VAR=VALUE ./configure`**.
- 26 de abril de 2002 [marcheerdink]: Capítulo 06 Findutils: añadido libexecdir=/usr/bin al comando make para solucionar una ruta equivocada a libexecdir en updatedb.
- 25 de abril de 2002 [gerard]: Capítulo 06 Glibc: añadida una nota en la que si quieres instalar manualmente algunas locales, en lugar de todas, primero se debe crear el directorio `/usr/lib/locale`.
- 21 de abril de 2002 [gerard & markh]: Actualizado a MAKEDEV−1.5

- 12 de abril de 2002 [markh]: Añadir el directorio entities/ al cvs y extraerlas de index.xml.
- 19 de abril de 2002 [marcheerdink]: Actualizado a los siguiente paquetes: bison−1.35, diffutils−2.8.1, texinfo−4.2, util−linux−2.11q
- 9 de abril de 2002 [marcheerdink]: Añadido −−disable−perl−regexp a las opciones del configure de grep para evitar que intente enlazarse con una librería pcre no existente.
- 8 de abril de 2002 [gerard]: Añadido el servidor alternativo http://ftp.de.linuxfromscratch.org (complementario del ftp://ftp.de).

Listas de correo y archivos

El servidor linuxfromscratch.org hospeda las siguientes listas de correo de acceso público:

- lfs−support
- lfs−dev
- lfs−announce
- lfs−book
- lfs−chat
- lfs−security
- alfs−discuss
- blfs−dev
- blfs−book
- blfs−support

lfs−support

La lista de correo lfs−support proporciona soporte a los usuarios que se están construyendo un sistema LFS como el descrito en el libro principal. Las solicitudes de ayuda para instalar software no incluido en el sistema base deben hacerse en la lista blfs−support.

lfs−dev

La lista de correo lfs−dev es para discutir el desarrllo del libro LFS.

lfs−announce

La lista lfs−announce es una lista moderada para aquellos que quieren las notiicaciones de nuevas versiones estables sin el alto tráfico de lfs−dev.

lfs−book

La lista lfs−book es usada para coordinar el mantenimiento del libro LFS. Su tráfico es básicamente mensajes de Bugzilla y modificaciones en el CVS. Es importante que todo el desarrollo de discusiones sobre cosas de interes para los usuarios del libro se haga en lfs−dev, no aquí.

lfs−chat

La lista lfs−chat es un lugar de encuentro comunitario. Es un sitio donde todo cabe, nada está fuera de lugar. Puedes discutir el precio de la cerveza o qué hardware comprar. Incluso las peleas entre GNU y BSD o entre Microsoft y Linux son aceptadas en lfs−chat.

lfs−security

La lista lfs−security es para discutir cuestiones relacionadas con la seguridad. Los avisos de vulnerabilidad, cuestiones sobre la configuración, paquetes relacionados con la seguridad y otras cosas para hacer seguro el sistema son apropiadas en esta lista.

alfs−discuss

La lista alfs−discuss es para discutir el desarrollo de ALFS.

blfs−dev

La lista blfs−dev es para discutir el desarrollo del libro BLFS.

blfs−book

La lista blfs−book es usada para coordinar el mantenimiento del libro BLFS. Su tráfico es básicamente mensajes de Bugzilla y modificaciones en el CVS. Es importante que todo el desarrollo de discusiones sobre cosas de interes para los usuarios del libro se haga en blfs−dev, no aquí.

blfs−support

La lista blfs−support es para el libro BLFS y más. En ella se incluyen peticiones de ayuda con el libro BLFS, peticiones de ayuda con paquetes que (todavía) no están ni en el libro LFS ni en el BLFS, y otras peticiones y discusiones relativas a software que podría instalarse en un sistema LFS. Esto no incluye tópicos como el precio de la cerveza, qué hardware comprar o las peleas de GNU contra BSD o Microsoft contra Linux. Esto pertenece a lfs−chat y, puesto que el tráfico en blfs−support es elevado, por favor respeta estrictamente esta regla.

Archivos de mensajes

Todas estas listas están archivadas y pueden verse en línea en http://archive.linuxfromscratch.org/mail−archives o descargarlas de http://ftp.linuxfromscratch.org/mail−archives o ftp://ftp.linuxfromscratch.org/mail−archives.

Cómo escribir en una lista

La dirección de envío a una lista tiene el formato *nombre_de_la_lista@linuxfromscratch.org* donde *nombre_de_la_lista* puede ser una de la listas mencionadas anteriormente. Ejemplos de las direcciones de envío son *lfs−support@linuxfromscratch.org* o *blfs−support@linuxfromscratch.org*.

Cómo suscribirse

Puedes suscribirte a cualquiera de las listas anteriormente mencionadas enviando un mensaje a listar@linuxfromscratch.org y escribiendo *subscribe nombre_de_la_lista* como asunto del mensaje.

Puedes suscribirte a múltiples listas con un solo mensaje. Esto se hace dejando el asunto en blanco y colocando todos los comandos en el cuerpo del mensaje. El mensaje tendrá este aspecto:

Para: listar@linuxfromscratch.org
Asunto:

subscribe lfs−dev
subscribe blfs−support
subscribe alfs−discuss

Después de enviar el mensaje, el programa Listar te enviará un mensaje solicitando la confirmación a la petición de suscripción. Después de enviar la confirmación, Listar te enviará un mensaje diciendo que has sido suscrito a la lista o listas y una introducción particular para cada lista.

Cómo desuscribirse

Para desuscribirte de una lista envía un mensaje a listar@linuxfromscratch.org y pon *unsubscribe nombre_de_la_lista* como asunto del mensaje.

Puedes desuscribirte de múltiples listas con un solo mensaje. Esto se hace dejando el asunto en blanco y colocando todos los comandos en el cuerpo del mensaje. El mensaje tendrá este aspecto:

Para: listar@linuxfromscratch.org
Asunto:

unsubscribe lfs−dev
unsubscribe blfs−support
unsubscribe alfs−discuss

Después de enviar el mensaje, el programa Listar te enviará un mensaje solicitando la confirmación a la petición de desuscripción. Después de enviar la confirmación, Listar te enviará un mensaje diciendo que has sido desuscrito de la lista o listas.

Otros modos de las listas

Para que el usuario active los distintos modos de uso debe enviar un mensaje a listar@linuxfromscratch.org. Los modos de uso se establecen poniendo el comando apropiado como asunto del mensaje.

Como su nombre implica, el comando *Set* indica que se activa un modo de uso. El comando *Unset* indica que se desactiva un modo de uso.

La palabra "nombre_de_la_lista" en los asuntos de ejemplo mostrados a continuación debe reemplazarse con el nombre de la lista a la que se desea aplicar ese modo de uso. Si se necesita activar en un sólo mensaje más de un modo de uso (en la misma lista o en varias listas) puede hacerse dejando el asunto en blanco y escribiendo los comandos en el cuerpo del mensaje.

Modo de resumen (Digest)

Activar: *set nombre_de_la_lista digest*
Desactivar: *unset nombre_de_la_lista digest*

Todas las listas tienen disponible el modo de resumen, que puede activarse después de que el usuario se suscriba a la lista. Entrar en el modo de resumen hace que pares de recibir mensajes individuales al ritmo que se van publicando y, en su lugar, recibirás un mensaje diario conteniendo todos los mensajes publicados durante ese día.

Hay otro modo de resumen llamado digest2. Cuando el usuario activa este modo de uso recibe el resumen diario, pero también continúa recibiendo los mensajes individuales. Para activar este modo de uso sustituye *digest* por *digest2* en el comando.

Vacaciones

Activar: *set nombre_de_la_lista vacation*
Desactivar: *unset nombre_de_la_lista vacation*

Si un usuario va a ausentarse un tiempo, o desea parar de recibir mensajes sin necesidad de desuscribirse, puede cambiar al modo de vacaciones. Esto tiene el mismo efecto que desuscribirse, pero sin tener que pasar por el proceso de desuscripción y posterior suscripción.

Servidores de noticias

Todas las listas de correo hospedadas en linuxfromscratch.org también son accesibles a través del servidor NNTP. Todos los mensajes publicados en una lista de correo son copiados en el grupo de noticias correspondiente y viceversa.

El servidor de noticias (news) a usar es *news.linuxfromscratch.org*.

FAQ

Si encuentras algún error, tienes preguntas o encuentras un error tipográfico en el libro, entonces consulta la página de las FAQ (Cuestiones Preguntadas Frecuentemente).

Versión en castellano: http://www.escomposlinux.org/lfs−es/faq.

Versión original en inglés: http://www.linuxfromscratch.org/faq/.

Información de contacto

Por favor, envía tus mensajes a las listas de correo. En el Capítulo 1 − Listas de correo y archivos tienes información sobre las listas de correo disponibles.

Si necesitas contactar directamente con Gerard Beekmans, manda un mensaje a gerard@linuxfromscratch.org

Capítulo 2. Información importante

Sobre $LFS

Por favor, lee con atención: en este libro la variable LFS se usará frecuentemente. $LFS deberá sustituirse en todo momento por el directorio en el que se monta la partición que contiene el sistema LFS. Cómo crear y dónde montar la partición se explicará con todo detalle en el Capítulo 4. Por ejemplo, supongamos que la partición LFS está montada en /mnt/lfs.

Si las instrucciones son ejecutar un comando como **`./configure --prefix=$LFS/static`**, en realidad debes ejecutar **`./configure --prefix=/mnt/lfs/static`**.

Es importante hacer esto donde quiera que aparezca, ya sea en comandos introducidos en un intérprete de comandos, o al crear o editar un archivo.

Una posible solución es establecer la variable de entorno LFS. De este modo $LFS puede introducirse literalmente, en lugar de substituirlo por /mnt/lfs. Esto se consigue ejecutando:

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

Ahora, cuando las instrucciones sean ejecutar un comando como **`./configure --prefix=$LFS/static`** puedes introducir eso literalmente. Tu intérprete de comandos substituirá $LFS con /mnt/lfs al procesar la línea de comando (es decir, cuando pulses Enter después de haber tecleado el comando).

Sobre los SBUs

Los SBUs son *Static Bash Units (Unidades de Bash Estático)* y son nuestro método para identificar cuanto tiempo se necesita para compilar un paquete. ¿Por qué no usamos tiempos normales como todo el mundo?

El mayor problema es que esos tiempos no pueden ser precisos, ni siquiera en parte. Puesto que muchas personas instalan LFS sobre muchos sistemas diferentes, el tiempo que se tarda en compilar algo varía demasiado. Un paquete puede tardar 20 minutos en un sistema, pero el mismo paquete puede tardar 3 días en otro sistema (esto no es una exageración). Así que en su lugar utilizamos el *Static Bash Unit* o *SBU*.

Funciona de esta forma: el primer paquete que compilas en el libro es Bash, en el Capítulo 5, y será enlazado estáticamente. El tiempo que se necesita para compilar este paquete será tomado como unidad base y llamado SBU. Todos los demás tiempos de compilación son relativos al tiempo necesario para instalar Bash. Por ejemplo, GCC−3.2 necesita unos 9.5 SBUs y está comprobado que este número es bastante preciso para muchos sistemas diferentes. Así que, multiplicando 9.5 por los segundos que has necesitado para instalar Bash (el valor SBU), obtendrás una buena aproximación del tiempo que tardará GCC en tu sistema.

Nota: Hemos visto que los SBUs no funcionan bien en máquinas basadas en SMP (Multi−Procesadores Simétricos). Por lo que todas las apuestas quedan en el aire si eres lo bastante afortunado para tener una disposición SMP.

Cómo buscar ayuda

Si tienes algún problema usando este libro, y tu problema no aparece en las FAQ (en castellano en http://www.escomposlinux.org/lfs−es/faq, y en inglés en http://www.linuxfromscratch.org/faq), encontrarás que la mayoría de la gente en el Internet Relay Chat (IRC) y en las listas de correo estará dispuesta a ayudarte. Puedes encontrar una relación de las listas de correo de LFS en Capítulo 1 − Listas de correo y archivos. Para facilitarnos la tarea de identificar y resolver tu problema, incluye toda la información relevante que sea posible en tu petición de ayuda.

Cosas que debes mencionar

Además de una breve explicación del problema que estás teniendo, debes incluir lo siguiente en tu petición:

- la versión del libro que estás usando (que es 4.1),
- la distrubición anfitrión, y su versión, que estás usando como base para crear el LFS,
- el paquete o la sección que te da problemas
- el mensaje de error exacto o los síntomas que aparecen,
- si te has desviado o no del libro.

(Ten en cuenta que decir que no has seguido las recomendaciones del libro no implica que no vayamos a ayudarte. Después de todo, la razón de ser de LFS es la posibilidad de elección. Simplemente nos ayudará a detectar otras posibles causas de tu problema)

Problemas de Configuración

Cuando algo vaya mal en la fase en que se ejecuta el guión configure, consulta las últimas líneas de `config.log`. Este fichero puede contener errores encontrados durante la configuración que no se muestran en pantalla. Incluye esas líneas relevantes si decides pedir ayuda.

Problemas de Compilación

Para ayudarnos a determinar la causa del problema, nos va a ser útil tanto la salida del terminal como el contenido de varios ficheros. Las salidas al terminal del guión configure y del comando make pueden ser útiles. No incluyas ciegamente todo el contenido pero, por otro lado, no incluyas demasiado poco. Por ejemplo, aquí hay una salida a terminal 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 de la sección anterior desde donde pone

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

hasta el final. Esto no nos basta 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.

Hay un artículo excelente sobre cómo buscar ayuda en Internet, escrito por Eric S. Raymond. Está disponible en http://www.tuxedo.org/~esr/faqs/smart-questions.html. Lee y sigue los consejos de este documento y tendrás muchas más posibilidades de obtener una respuesta, y también de que obtengas la ayuda que necesitas.

II. Parte II – Preparativos para la construcción

Índice

3. Preparación de una nueva partición
4. Paquetes que hay que descargar
5. Preparación del sistema LFS

Capítulo 3. Preparación de una nueva partición

Introducción

En este capítulo se preparará la partición que contendrá el sistema LFS. Crearemos la partición, haremos un sistema de ficheros en ella, y la montaremos.

Crear una nueva partición

Para construir nuestro nuevo sistema Linux necesitaremos algún espacio: una partición de disco vacía. Si no tienes una partición libre, y no tienes sitio en ninguno de tus discos duros para crear una, entonces puedes construir LFS en la misma partición en la que tienes instalada tu distribución actual. Este proceso no es recomendable para tu primera instalación del LFS, pero si andas escaso de espacio en el disco y te sientes valiente, hecha un vistazo a la receta http://www.escomposlinux.org/lfs−es/recetas/lfs_next_to_existing_systems.html (la versión original en inglés se encuentra en http://hints.linuxfromscratch.org/hints/lfs_next_to_existing_systems.txt).

Para un sistema mínimo necesitas una partición de 1 GB más o menos. Esto es suficiente para almacenar todos los archivos de código fuente y compilar todos los paquetes. Pero, si piensas usar tu sistema LFS como tu sistema Linux primario, seguramente querrás instalar software adicional y necesitarás más espacio, posiblemente sobre 2 o 3 GB.

Como casi nunca tenemos suficiente memoria RAM en nuestra máquina, es buena idea utilizar una paqueña partición como espacio de intercambo (swap) −− este espacio lo usa el núcleo para almacenar los datos menos usados y hacer sitio en memoria para las cosas urgentes. La partición de intercambio para tu sistema LFS puede ser la misma que la de tu sistema anfitrión, por lo que no tienes que crear otra si tu sistema anfitrión ya utiliza una partición de intercambio.

Inicia el programa `cfdisk` 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 la página de manual de `cfdisk` si todavía no sabes cómo usar el programa.

Recuerda la denominación de tu nueva partición −− será algo como `hda5`. En este libro nos referiremos a ella como la partición LFS. Si (ahora) tienes además una partición de intercambio, recuerda también su denominación. Estos nombres se necesitarán posteriormente para el fichero `/etc/fstab`.

Crear un sistema de ficheros en la nueva partición

Una vez que se ha creado la partición, podemos crear un sistema de ficheros en esa partición. El sistema de ficheros más usado en el mundo de Linux es el "segundo sistema de ficheros extendido (second extended file system)", conocido como ext2. Pero, con la gran capacidad de los discos duros actuales, los llamados sistemas de ficheros con registro de transaciones (journaling) se están haciedo muy populares. Aquí crearemos un sistema de ficheros ext2, sin embargo encontrarás las instrucciones de construcción para otros sistemas de ficheros en http://www.escomposlinux.org/lfs−es/blfs−es−CVS/postlfs/filesystems.html (la versión original la tienes en http://beyond.linuxfromscratch.org/view/cvs/postlfs/filesystems.html).

Para crear un sistema de ficheros ext2 en la partición LFS ejecuta lo siguiente:

```
mke2fs /dev/xxx
```

Sustituye `xxx` con el nombre de la partición LFS (algo como `hda5`).

Si creas una (nueva) partición de intercambio, también necesitas inicializarla como partición de intercambio (también conocido como formatearla, como hiciste anteriormente con **`mke2fs`**) ejecutando:

```
mkswap /dev/yyy
```

Sustituye `yyy` con el nombre de la partición de intercambio.

Montar la nueva partición

Ahora que hemos creado un sistema de ficheros, queremos poder acceder a la partición. Para esto necesitamos montarla y tener elegido un punto de montaje. En este libro asumimos que el sistema de ficheros se monta en `/mnt/lfs`, pero no importa el directorio que elijas.

Elije un punto de montaje y asignale la variable de entorno LFS ejecutando:

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

Después crea el punto de montaje y monta el sistema de ficheros LFS ejecutando:

```
mkdir -p $LFS &&
mount /dev/xxx $LFS
```

Sustituye `xxx` con la designación de la partición LFS.

Si tienes decidido usar mútiples particiones para LFS (digamos que una para `/` y otra para `/usr`), montalas de esta forma:

```
mkdir -p $LFS &&
mount /dev/xxx $LFS &&
mkdir $LFS/usr &&
mount /dev/yyy $LFS/usr
```

Por supuesto, sustituye `xxx` y `yyy` con el nombre de partición apropiado.

Ahora que nos hemos hecho un sitio en el que trabajar, estamos preparados para descargar los paquetes.

Capítulo 4. Paquetes que hay que descargar

Introducción

A continuación sigue una lista con los paquetes que necesitas descargar para construir un sistema Linux básico. Los números de versión listados corresponden a versiones del software que se sabe que funcionan, y este libro se basa en ellos. Eres libre de probar nuevas versiones, pero si tienes problemas con esas versiones, por favor, prueba con la versión recomendada antes de enviar informes de error.

Todas las URLs, cuando es posible, apuntan a la página del proyecto en Freshmeat.net. Las páginas de Freshmeat proporcionan un acceso fácil a los sitios oficiales de descarga, así como a los sitios web del proyecto, listas de correo, FAQs, Historiales de modificaciones y más cosas.

Para facilitarte la tarea, al principio de la lista tienes un enlace a un fichero que puedes usar con el programa wget. Con este fichero y el programa **`wget`** te será fácil descargar todos los ficheros de una vez, en lugar de descargar individualmente todos y cada uno de ellos de forma manual.

Paquetes que hay que descargar

Puedes descargar el guión para **`wget`** y descargar todos los paquetes automáticamente:

[Crear el guión de wget y subirlo al servidor]

O descargar los paquetes individualmente:

Autoconf (2.57) – 792 KB:
http://freshmeat.net/projects/autoconf/

Automake (1.7.2) – 518 KB:
http://freshmeat.net/projects/automake/

Bash (2.05a) – 1,766 KB:
http://freshmeat.net/projects/gnubash/

Bin86 (0.16.3) – 142 KB:
http://freshmeat.net/projects/bin86/

Binutils (2.13.2) – 9,534 KB:
http://freshmeat.net/projects/binutils/

Bison (1.875) – 796 KB:
http://freshmeat.net/projects/bison/

Bzip2 (1.0.2) – 650 KB:
http://freshmeat.net/projects/bzip2/

Diffutils (2.8.1) – 762 KB:

http://freshmeat.net/projects/diffutils/

E2fsprogs (1.32) − 2,827 KB:
http://freshmeat.net/projects/e2fsprogs/

Ed (0.2) − 182 KB:
http://freshmeat.net/projects/ed/

Parche para Ed (0.2) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/ed−0.2.patch
http://downloads.linuxfromscratch.org/ed−0.2.patch

File (3.39) − 177 KB:
http://freshmeat.net/projects/file/

Fileutils (4.1) − 1,770 KB:
http://freshmeat.net/projects/gnufileutils/

Parche para Fileutils (4.1) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/fileutils−4.1.patch
http://downloads.linuxfromscratch.org/fileutils−4.1.patch

Findutils (4.1) − 288 KB:
http://freshmeat.net/projects/findutils/

Parche para Findutils (4.1) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/findutils−4.1.patch
http://downloads.linuxfromscratch.org/findutils−4.1.patch

Parche para la violación de segmento en Findutils(4.1) − 1 KB:
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/findutils−4.1−segfault.patch
http://downloads.linuxfromscratch.org/findutils−4.1−segfault.patch

Flex (2.5.4a) − 372 KB:
http://freshmeat.net/projects/flex/

Gawk (3.1.1) − 1,831 KB:
http://freshmeat.net/projects/gnuawk/

Parche para Gawk (3.1.1−3) − 1 KB:
http://downloads.linuxfromscratch.org/gawk−3.1.1−3.patch

GCC−core (3.2.1) − 13,140 KB:
http://freshmeat.net/projects/gcc/

GCC−g++ (3.2.1) − 2520 KB:
http://freshmeat.net/projects/gcc/

Gettext (0.11.5) − 3,637 KB:
http://freshmeat.net/projects/gettext/

Glibc (2.3.1) – 17,463 KB:
http://freshmeat.net/projects/glibc/

Parche Root/Perl para Glibc: (2.3.1) – 1 KB:
http://downloads.linuxfromscratch.org/glibc-2.3.1-root-perl.patch

Parche Libnss para Glibc: (2.3.1) – 1 KB:
http://downloads.linuxfromscratch.org/glibc-2.3.1-libnss.patch

Glibc-linuxthreads (2.3.1) – 238 KB:
http://freshmeat.net/projects/glibc/

Grep (2.5) – 545 KB:
http://freshmeat.net/projects/grep/

Groff (1.18.1) – 2,198 KB:
http://freshmeat.net/projects/groff/

Gzip (1.2.4a) – 216 KB:
http://freshmeat.net/projects/gzip/

Parche para Gzip (1.2.4b) – 1 KB:
ftp://ftp.linuxfromscratch.org/lfs-packages/cvs/gzip-1.2.4b.patch
http://downloads.linuxfromscratch.org/gzip-1.2.4b.patch

Kbd (1.08) – 801 KB:
http://freshmeat.net/projects/kbd/

Parche para Kbd (1.08) – 1 KB:
ftp://ftp.linuxfromscratch.org/lfs-packages/cvs/kbd-1.08.patch
http://downloads.linuxfromscratch.org/kbd-1.08.patch

Less (378) – 239 KB:
http://freshmeat.net/projects/less/

LFS-Bootscripts (1.11) – 25 KB:
http://downloads.linuxfromscratch.org/lfs-bootscripts-1.11.tar.bz2

Libtool (1.4.3) – 1137 KB:
http://freshmeat.net/projects/libtool/

Lilo (22.2) – 343 KB:
http://freshmeat.net/projects/lilo/

Linux (2.4.20) – 26,778 KB:
http://freshmeat.net/projects/linux/

M4 (1.4) – 310 KB:
http://freshmeat.net/projects/gnum4/

Make (3.80) – 899 KB:
http://freshmeat.net/projects/gnumake

MAKEDEV (1.7) – 8 KB:
ftp://ftp.linuxfromscratch.org/lfs-packages/cvs/MAKEDEV-1.7.bz2
http://downloads.linuxfromscratch.org/MAKEDEV-1.7.bz2

Man (1.5k) – 189 KB:
http://freshmeat.net/projects/man/

Parche 80Cols para Man (1.5k) – 1 KB:
http://downloads.linuxfromscratch.org/man-1.5k-80cols.patch

Parche Manpath para Man (1.5k) – 1 KB:
http://downloads.linuxfromscratch.org/man-1.5k-manpath.patch

Parche Pager para Man (1.5k) – 1 KB:
http://downloads.linuxfromscratch.org/man-1.5k-pager.patch

Man-pages (1.54) – 583 KB:
http://freshmeat.net/projects/man-pages/

Modutils (2.4.22) – 214 KB:
http://freshmeat.net/projects/modutils/

Ncurses (5.3) – 2,019 KB:
http://freshmeat.net/projects/ncurses/

Netkit-base (0.17) – 55 KB:
http://freshmeat.net/projects/netkit/

Net-tools (1.60) – 194 KB:
http://freshmeat.net/projects/net-tools/

Patch (2.5.4) – 183 KB:
http://freshmeat.net/projects/patch/

Perl (5.8.0) – 10,765 KB:
http://freshmeat.net/projects/perl/

Procinfo (18) – 24 KB:
http://freshmeat.net/projects/procinfo/

Procps (3.1.5) – 233 KB:
http://freshmeat.net/projects/procps/

Parche para Procps (3.1.5) – 1 KB:
http://downloads.linuxfromscratch.org/procps-3.1.5.patch

Psmisc (21.2) – 253 KB:

http://freshmeat.net/projects/psmisc/

Sed (4.0.5) – 665 KB:
http://freshmeat.net/projects/sed/

Shadow (4.0.3) – 1030 KB:
http://freshmeat.net/projects/shadow/

Sh–utils (2.0) – 1214 KB:
http://freshmeat.net/projects/sh–utils/

Parche Hostname para Sh–utils (2.0–hostname) – 1 KB:
ftp://ftp.linuxfromscratch.org/lfs–packages/cvs/sh–utils–2.0–hostname.patch
http://downloads.linuxfromscratch.org/sh–utils–2.0–hostname.patch

Parche para Sh–utils (2.0) – 1 KB:
ftp://ftp.linuxfromscratch.org/lfs–packages/cvs/sh–utils–2.0.patch
http://downloads.linuxfromscratch.org/sh–utils–2.0.patch

Sysklogd (1.4.1) – 80 KB:
http://freshmeat.net/projects/sysklogd/

Sysvinit (2.84) – 89 KB:
http://freshmeat.net/projects/sysvinit/

Tar (1.13) – 1028 KB:
http://freshmeat.net/projects/tar/

Parche para Tar (1.13) – 1 KB:
ftp://ftp.linuxfromscratch.org/lfs–packages/cvs/tar–1.13.patch
http://downloads.linuxfromscratch.org/tar–1.13.patch

Texinfo (4.3) – 1,254 KB:
http://freshmeat.net/projects/texinfo/

Textutils (2.1) – 1,847 KB:
http://freshmeat.net/projects/textutils/

Util–linux (2.11y) – 1,777 KB:
http://freshmeat.net/projects/util–linux/

Vim (6.1) – 2,822 KB:
http://freshmeat.net/projects/vim/

Parche para Vim (6.1) – 1 KB:
ftp://ftp.linuxfromscratch.org/lfs–packages/cvs/vim–6.1.patch
http://downloads.linuxfromscratch.org/vim–6.1.patch

Zlib (1.1.4) – 144 KB:
http://freshmeat.net/projects/zlib/

Tamaño total de todos los paquetes: 99,270 KB (96.96 MB)

Capítulo 5. Preparación del sistema LFS

Introducción

En este capítulo compilaremos e instalaremos un sistema Linux mínimo. Este sistema contendrá sólo las herramientas necesarias para poder iniciar la construcción del sistema LFS definitivo en el siguiente capítulo.

Los ficheros que se compilen en este capítulo se instalarán bajo el directorio `$LFS/static`, para mantenerlos separados de los ficheros que se instalen en el siguiente capítulo. Como todo lo hecho aquí es solamente temporal, no queremos que estos ficheros contaminen el futuro sistema LFS.

La clave para aprender qué es lo que hace funcionar un sistema Linux es saber exactamente para qué se usa cada paquete, y por qué el usuario o el sistema lo necesita. Por esta razón se facilita una descripción corta del contenido de cada paquete a continuación de las instrucciones de instalación.

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 este como en el siguiente capítulo, pero a veces sólo es necesario en uno de ellos. Por tanto, no te preocupes cuando parezca que hemos olvidado las instrucciones para uno de los parches descargados.

Durante la instalación de bastantes paquetes verás aparecer en pantalla todo tipo de avisos (warnings). Esto es normal y puedes ignorarlos tranquilamente. No son más que eso, avisos; la mayoría debidos a un uso inapropiado, pero no ilegal, de la sintaxis de C o C++. Se debe a que el estándar C cambia con frecuencia y algunos paquetes todavía usan el estándar antiguo, lo que no es realmente un problema.

Antes de continuar, asegúrate de que la variable de entorno LFS tiene el valor correcto (si es que has decidido utilizarla) ejecutando el siguiente comando:

```
echo $LFS
```

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

¿Por qué usamos enlazado estático?

La mayoría de los pogramas deben de realizar, dentro de sus tareas específicas, muchas otras operaciones comúnes y triviales, como escribir en memoria, buscar directorios, abrir y cerrar ficheros, leerlos y escribirlos, manejo de cadenas, reconocimiento de patrones, aritmética y más. En lugar de obligar a todos los programas a reinventar la rueda, el sistema GNU proporciona todas estas funciones básicas preparadas para su uso en librerías. La mayor librería de cualquier sistema Linux es `glibc`. Para tener una idea de lo que contiene, mira el fichero `glibc/index.html` que se encuentra en algún lugar de tu sistema anfitrión.

Hay dos formas de enlazar las funciones de una librería en un programa que las utilice: estáticamente o dinámicamente. Cuando un programa se enlaza estáticamente el código de las funciones utilizadas se incluye en el ejecutable, resultando un programa algo abultado. Cuando un programa se enlaza dinámicamente, lo que se incluye es una referencia al enlazador, el nombre de la librería y el nombre de la función, resultando un ejecutable mucho más pequeño. Este ejecutable tiene la desventaja de ser algo más lento que uno enlazado estáticamente, pues al enlazarse en tiempo de ejecución se consume algo de tiempo.

Aparte de esta pequeña desventaja, el enlazado dinámico tiene dos ventajas mayores sobre el enlazado estático. Primero, sólo necesitas una copia del código ejecutable de la librería en tu disco duro, en vez de tener muchas copias del mismo código en un montón de programas, salvando espacio en el disco. En segundo lugar, cuando varios programas usan las mismas funciones de las librerías al mismo tiempo, sólo se requiere la carga de una copia del código de las funciones, salvando espacio en memoria.

Hoy en día salvar algunos megabytes de espacio puede que no parezca mucho, pero hace muchas lunas, cuando los discos se median en megabytes y la memoria en kilobytes, este ahorro era sustancial. Significaba ser capaz de tener varios programas en memoria al mísmo tiempo y contener un sistema Unix completo en unos pocos volúmenes de discos.

Una tercera ventaja, aunque menor, del enlazado dinámico es que cuando una función de una librería tiene una correción de un error, o es mejorada, sólo necesitas recompilar esta librería, en lugar de tener que recompilar todos los programas que usan esta función mejorada.

En resumen, podemos decir que el enlazado dinámico consume tiempo de ejecución en vez de espacio en memoria, espacio en disco y tiempo de recompilación.

Pero si el enlazado dinámico salva tanto espacio, ¿por qué entonces vamos a enlazar todos los programas de este capítulo estáticamente?. La razón es que no vamos a compilar aquí una `glibc` temporal. Y evitamos hacer esto simplemente para ahorrar algo de tiempo, sobre unos 14 SBUs. Otra razón es que la versión de Glibc del sistema LFS puede que no sea compatible con la Glibc del sistema anfitrión. Las aplicaciones compiladas con la Glibc de tu sistema anfitrión puede que no se ejecuten adecuadamente (o en absoluto) en el sistema LFS.

Esto significa que las herramientas compiladas en este capítulo deberán ser auto−contenidas, debido a que cuando posteriormente hagamos chroot a la partición LFS la librería GNU no estará disponible. Esto es por lo que usamos los modificadores **`-static`**, **`--enable-static-link`**, y **`--disable-shared`** a lo largo de este capítulo, para asegurarnos de que todos los ejecutables están enlazados estáticamente. Cuando vayamos al siguiente capítulo, casi la primera cosa que haremos es construir `glibc`, el conjunto principal de librerías del sistema. Una vez hecho, podemos enlazar todos los demás programas dinámicamente (incluidos los instalados estáticamente en este capítulo) y sacaremos provecho de las oportunidades de ahorro de espacio.

Creación del directorio $LFS/static

Todos los programas compilados en este capítulo serán instalados bajo `$LFS/static` para mantenerlos separados de los programas compilados en el siguiente capítulo. Los programas compilados aquí son sólo herramientas temporales y no formarán parte del sistema LFS final, por lo que manteniendolos en un directorio aparte posteriormente podremos quitarlos más facilmente. Crea el directorio requerido ejecutando lo siguiente:

```
mkdir $LFS/static
```

Añadir el usuario lfs

Si trabajas como *root* durante el Capítulo 5, tu sistema anfitrion puede resultar dañado por un simple error. Recomendamos que construyas los paquetes del Capítulo 5 como un usuario sin privilegios. Puedes usar tu propio nombre de usuario, pero para asegurar un entorno de construcción limpio crearemos un nuevo usuario: *lfs*. Como *root*, ejecuta el siguiente comando para añadir el nuevo usuario:

```
useradd -s /bin/bash -m lfs
passwd lfs
```

Para conceder la propiedad del directorio `$LFS/static` al usuario *lfs*, ejecuta el comando:

```
chown lfs $LFS/static
```

Ahora, entra como usuario *lfs*. Esto puede hacer mediante una consola virtual, un administrador de sesión o con el comando de sustitución de usuario:

```
su - lfs
```

El "**-**" le indica a **su** que inicie un interprete de comandos nuevo y limpio.

Configuración del entorno

Ahora que has entrado al sistema como usuario *lfs*, ejecuta el siguiente comando para establecer un buen entorno de trabajo:

```
cat > ~/.bash_profile << "EOF"
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
CC="gcc -s"
export LFS LC_ALL CC
EOF
source ~/.bash_profile
```

Este perfil establece la máscara a 022, así los ficheros y directorios recién creados tendrán los permisos apropiados. Para ser más especificos, sólo el propietario del fichero tendrá permisos de escritura en los nuevos ficheros y directorios. Otros usuarios del sistema tendrán permisos de lectura, y permisos para entrar a los directorios. Es recomendable mantener este ajuste a lo largo de tu instalación del LFS.

La variable LFS deberas establecerla con el punto de montaje que hayas elegido.

La variable LC_ALL controla la localización de ciertos programas, haciendo que sus mensajes sigan las convenciones para un determinado país. Si tu sistema anfitrión utiliza una versión de *glibc* más antigua que 2.2.4, tener LC_ALL establecida a algo diferente a "C" o "POSIX" durante este capítulo puede causar problemas si sales del entorno chroot e intentas regresar más tarde. Estableciendo LC_ALL a "POSIX" ("C" es un alias para "POSIX") nos aseguramos de que todo funcionará como se espera en el entorno chroot.

CC es una variable que establecemos para evitar que los símbolos de depuración sean compilados dentro de nuestros paquetes estáticos. Al omitir estos símbolos durante la fase de compilación salvamos espacio en el disco duro y reducimos el tiempo de construcción.

Ahora estamos preparados para comenzar la construcción de las herramientas temporales que nos ayudarán en capítulos posteriores.

Instalación de Bash–2.05a

```
Estimación del tiempo de construcción:       1 SBU
Estimación del espacio necesario en disco:  24 MB
```

Contenido de Bash

Última versión comprobada: 2.05a.

bash es la "Bourne–Again SHell", que es un completo intérprete de comandos usado ampliamente en sistemas Unix. El programa bash lee de la entrada estándar (el teclado). Un usuario escribe algo y el programa evalúa lo que ha escrito y hace algo con ello, como lanzar un programa.

Bash instala lo siguiente:

Programas

bash, sh (enlace a bash) y bashbug

Dependencias de instalación de Bash

Última versión comprobada: 2.05a.

Bash: bash, sh
Binutils: ar, as, ld, ranlib, size
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Gawk: awk
Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install–info
Textutils: cat, tr, uniq

Instalación de Bash

Antes de intentar instalar Bash tienes que asegurarte de que tu distribución tiene los archivos `/usr/lib/libcurses.a` y `/usr/lib/libncurses.a`. Si tu distribución original es otro sistema LFS y seguiste las instrucciones del libro al pie de la letra, todos estos ficheros existirán.

Si no existe ninguno de los dos ficheros, debes instalar el paquete de desarrollo de la librería Ncurses. Este paquete suele tener un nombre parecido a *ncurses–dev* o *ncurses–static*. Si este paquete ya está instalado o

acabas de instalarlo, comprueba los dos ficheros de nuevo. A menudo el fichero `libcurses.a` puede faltar (todavía). Si es ese el caso, entonces crea un enlace simbólico `libcurses.a`, ejecutando como usuario *root* el siguiente comando:

```
ln -s libncurses.a
/usr/lib/libcurses.a
```

Ahora podemos comenzar de verdad. Prepara Bash para su compilación ejecutando el siguiente comando:

```
./configure --enable-static-link \
    --prefix=$LFS/static
--with-curses
```

El significado de las opciones de configuración es:

- **`--enable-static-link:`** Esta opción hace que el programa **`bash`** sea enlazado estáticamente.
- **`--prefix=$LFS/static:`** Esta opción de configure instala todos los ficheros de Bash bajo el directorio `$LFS/static`, que pasa a ser el directorio `/static` cuando entramos en el entorno chroot o reiniciamos LFS.
- **`--with-curses:`** Esto provoca que `bash` se enlace con la librería curses en lugar de la librería por defecto, termcap, que ha quedando obsoleta y desaparecerá. Advierte que, en casi todas las distribuciones, la librería curses es suministrada por el paquete Ncurses (por lo que en realidad enlazamos con la librería ncurses).

  No es estrictamente necesario enlazar bash con libncurses (podría, por el momento, enlazarse con una librería estática termcap y no pasaría nada porque, de todas formas, instalaremos de nuevo Bash en el Capítulo 6, donde usaremos libncurses), pero es una buena forma de comprobar que el paquete Ncurses se ha instalado correctamente. Si no es así, podrías tener problemas más adelante en este capítulo, cuando instales el paquete Texinfo. Ese paquete necesita ncurses y no se puede utilizar termcap en ese caso.

Ahora podemos continuar compilando Bash:

```
make
```

Y terminar la instalación instalando Bash:

```
make install
```

Si al terminar la etapa **`make install`** aparecen unas líneas como estas:

```
install-info: unknown option
`--dir-file=/mnt/lfs/usr/info/dir'
usage: install-info [--version] [--help] [--debug] [--maxwidth=nnn]
        [--section regexp title] [--infodir=xxx] [--align=nnn]
        [--calign=nnn] [--quiet] [--menuentry=xxx]
        [--info-dir=xxx]
        [--keep-old] [--description=xxx] [--test]
        [--remove] [--] filename
make[1]: *** [install] Error 1
make[1]: Leaving directory `/mnt/lfs/usr/src/bash-2.05a/doc'
make: [install] Error 2 (ignored)
```

entonces, probablemente, estás utilizando Debian–2.2 (potato) y tienes una versión antigua del paquete texinfo, por lo que no podrás instalar las páginas info por el momento. Este error no es grave en absoluto: las páginas info se instalarán cuando volvamos a compilar bash dinámicamente en el Capítulo 6, así que puedes ignorarlo. Se ha informado de que la versión actual de Debian (3.0, también conocida como Woody) no tiene este problema.

Instalación de Binutils–2.13.2

```
Estimación del tiempo de construcción:       2.05 SBU
Estimación del espacio necesario en disco:  160 MB
```

Contenido de Binutils

Última versión comprobada: 2.12.1.

Binutils es una colección de herramientas para el desarrollo de software que contiene un enlazador, un ensamblador y otras utilidades para trabajar con ficheros de objetos y archivos.

Binutils instala lo siguiente:

Programas

addr2line, ar, as, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings y strip

Librerías

libbfd.[a,so] y libopcodes.[a,so]

Dependencias de instalación de Binutils

Última versión comprobada: 2.11.2.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, rmdir, touch
Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, true, uname
Texinfo: install–info, makeinfo

Textutils: cat, sort, tr, uniq

Instalación de Binutils

Se sabe que este programa se comporta mal si cambias sus parámetros de optimización (incluyendo las opciones −march y −mcpu). Por esta razón, si tienes definida cualquier variable de entorno que pueda sobreescribir las optimizaciones por defecto, como CFLAGS o CXXFLAGS, recomendamos quitarlas o modificarlas cuando construyas Binutils.

La documentación para la instalación de Binutil recomienda construir Binutils en un directorio dedicado fuera del árbol de las fuentes:

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

Seguidamente, prepara Binutils para su compilación:

```
../binutils-2.13.2/configure
--prefix=$LFS/static --disable-nls
```

El significado de la (nueva) opción de configure es:

- **`--disable-nls:`** Esta opción desactiva la internacionalización (también conocida como i18n). No la necesitamos para los programas estáticos, y nls a menudo causa problemas en el enlazado estático.

Continúa compilando el paquete:

```
make LDFLAGS="-all-static"
```

El significado de la opción de make es:

- **`LDFLAGS="-all-static":`** Así es como le decimos a Binutils que todos los programas deben enlazarse estáticamente. Establecer la variable *LDFLAGS* es la forma común de especificar que queremos que se haga un enlazado estático. Sin embargo, su valor y el modo de establecerlo no es siempre el mismo. En los paquetes que quedan por instalar verás que hay diferentes formas de establecer la variable *LDFLAGS*.

Y termina instalando el paquete:

```
make install
```

Instalación de Bzip2–1.0.2

```
Estimación del tiempo de construcción:       0.07 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Bzip2

Última versión comprobada: 1.0.2

Bzip2 es un compresor de ficheros por ordenación de bloques que, generalmente, consige una mejor compresión que el tradicional **gzip**.

Bzip2 instala lo siguiente:

Programas

bunzip2 (enlace a bzip2), bzcat (enlace a bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless y bzmore

Librerías

libbz2.a, libbz2.so (enlace a libbz2.so.1.0), libbz2.so.1.0 (enlace a libbz2.so.1.0.2) y libbz2.so.1.0.2

Dependencias de instalación de Bzip2

Última versión comprobada: 1.0.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Fileutils: cp, ln, rm
Gcc: cc1, collect2, cpp0, gcc
Make: make

Instalación de Bzip2

Puesto que el paquete Bzip2 no tiene un guión configure, no podemos prepararlo para su compilación. En lugar de esto ejecutamos simplemente el programa **make** e iniciamos la compilación, cambiando algunas variables para ajustar nuestro entorno:

```
make CC="gcc -static -s"
```

El significado de la opción de make es:

- **CC="gcc -static -s":** El paquete Bzip2 no reconoce la variable *LDFLAGS*, por lo que en su lugar establecemos la variable *CC* que define el compilador a usar. La opción *-static* le indica al compilador que enlace estáticamente todos los programas.

Y terminamos instalando el paquete:

```
make PREFIX=$LFS/static
install
```

El significado de la opción de make es:

- **`PREFIX=$LFS/static:`** en vez de pasarle *−−prefix=$LFS/static* al guión configure, establecemos la variable *PREFIX* para conseguir el mismo resultado (ya que no hay presente un guión configure).

Instalación de Diffutils–2.8.1

```
Estimación del tiempo de construcción:       0.39 SBU
Estimación del espacio rquerido en disco:  10 MB
```

Contenido de Diffutils

Última versión comprobada: 2.8.1.

Los programas de este paquete te muestran las diferencias entre dos ficheros o directorios. Es muy común usarlos para crear parches de software.

Diffutils instala lo siguientes:

Programas

cmp, diff, diff3 y sdiff

Dependencias de instalación de Diffutils

Última versión comprobada: 2.7.

Bash: sh
Binutils: ld, as
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: date, hostname
Textutils: cat, tr

Instalación de Diffutils

Prepara Diffutils para su compilación:

```
LDFLAGS="-static"
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static
--disable-nls
```

El significado de las opciones de configure es:

- **`LDFLAGS="-static":`** Esta es una de las formas comunes de indicarle a un paquete que todos los programas deben enlazarse estáticamente. De este modo se establece la variable de entorno *LDFLAGS*, pero sólo dentro del subintérprete de comandos en el que se ejecuta el guión `configure`. Cuando **`configure`** ha hecho su trabajo, la variable de entorno *LDFLAGS* desaparece y los ficheros `Makefile` contienen localmente esta variable.
- **`CPPFLAGS=-Dre_max_failures=re_max_failures2:`** La variable *CPPFLAGS* es leída por el programa cpp (PreProcesador de C). El valor de esta variable le dice al preprocesador que cambie cada aparición de *re_max_failures* que encuentre por *re_max_failures2* antes de enviar el código fuente al compilador para su compilación. Este paquete tiene problemas en el enlazado estático en sistemas que ejecutan una versión antigua de Glibc, y esta sustitución soluciona ese problema.

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Fileutils–4.1

```
Estimación del tiempo de construcción:       0.94 SBU
Estimación del espacio necesario en disco:  40 MB
```

Contenido de Fileutils

Última versión comprobada: 4.1.

Fileutils es un paquete que contiene los programas básicos para la manipulación de ficheros. Incluye programas para listar y crear directorios, actualizar las marcas de fechas, cambiar los permisos y más.

Fileutils instala lo siguiente:

Programas

chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, shred, sync, touch y vdir

Dependencias de instalación de Fileutils

Última versión comprobada: 4.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, tr

Instalación de Fileutils

Hay un error documentado en la función atexit() de glibc−2.2.3 cuando ésta fue compilada con gcc−2.95.3. Este error sólo se da en algunos sistemas (más que nada en AMD, pero no exclusivamente). Dicho error causa violaciones de segmento en fileutils−1.4 compilado estáticamente. Este parche hace que se llame a la función on_exit() en lugar de a atexit()

Ten en cuenta que, en algunos casos, utilizar este parche provocará que no se pueda compilar el paquete, incluso cuando tu sistema tenga un procesador AMD y una librería Glibc−2.2.3 (o superior) instalada. Si ese es tu caso, necesitarás borrar el directorio fileutils−4.1 y desempaquetarlo de nuevo antes de continuar. Creemos que esto puede pasar cuando tu distribución ha alterado de alguna forma la librería Glibc−2.2.3, pero desconocemos los detalles exactos.

Para reparar este paquete de forma que compile adecuadamente en máquinas AMD/Glibc−2.2.3, ejecuta el siguiente comando. *NO* intentes utilizar este arreglo si no tienes la librería Glibc−2.2.3 instalada. Es más que probable que provoque todo tipo de problemas de compilación.

```
patch -Np1 -i
../fileutils-4.1.patch
```

Prepara el paquete para su compilación:

```
LDFLAGS="-static" \
    ./configure --disable-nls
--prefix=$LFS/static
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Una vez hayas instalado Fileutils, puedes comprobar si se ha evitado el problema de la violación de segmento ejecutando **`$LFS/static/bin/ls`**. Si esto funciona, entonces está solucionado. Si no, deberás rehacer la instalación aplicando el parche si no lo usaste, o sin él, si en efecto lo utilizaste.

Instalación de Findutils–4.1

```
Estimación del tiempo de construcción:       0.12 SBU
Estimación del espacio necesario en disco:  8 MB
```

Contenido de Findutils

Última versión comprobada: 4.1.

El paquete Findutils contiene pogramas para encontrar ficheros, tanto al vuelo (haciendo una búsqueda recursiva en vivo a través de los directorios y mostrando sólo los ficheros que cumplan las especificaciones) o mediante una búsqueda a través de una base de datos.

Findutils instala lo siguiente:

Programas

bigram, code, find, frcode, locate, updatedb y xargs

Dependencias de instalación de Findutils

Última versión comprobada: 4.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Grep: egrep, grep
Gcc: cc1, collect2, cpp0, gcc
Make: make
Patch: patch
Sed: sed
Sh−utils: basename, date, echo, hostname
Textutils: cat, tr

Instalación de Findutils

El paquete Findutils es bastante antiguo y tiene problemas al compilarlo con versiones recientes de Glibc (Glibc−2.0.x y superiores). Necesitas aplicar el parche que soluciona estos conflictos:

- Findutils declara una función llamada basename, pero este nombre de función ya es utilizado por las librerías estandar de C. Este parche cambia la función basename de Findutils para que pase a llamarse basename2.
- Findutils usa las librerías de una forma incorrecta. Declara funciones de librería para decirle al compilador cuál será el nombre de la función cuando las librerías sean enlazadas. Dicha sintaxis es inválida, provocando advertencias y errores del compilador. Este parche elimina la sintaxis incorrecta.
- Todos los paquetes GNU deben usar una macro llamada _GNU_SOURCE. Esta macro hace que se incluyan las declaraciones de las funciones de extensión de la librería GNU para que el compilador pueda detectar con facilidad los conflictos de nombres entre funciones. Este parche agrega la macro al código fuente.

```
patch −Np1 −i
../findutils−4.1.patch
```

Prepara Findutils para su compilación:

```
CPPFLAGS="−Dre_max_failures=re_max_failures2" \
    LDFLAGS="−static" ./configure
−−prefix=$LFS/static
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Gawk−3.1.1

```
Estimación del tiempo de construcción:       0.39 SBU
Estimación del espacio necesario en disco:  17 MB
```

Contenido de Gawk

Última versión comprobada: 3.1.1.

Gawk es uns implementación de awk utilizada para manipular ficheros de texto.

Gawk instala lo siguiente:

Programas

awk, gawk, gawk–3.1.1, grcat, igawk, pgawk, pgawk–3.1.1, pwcat

Dependencias de instalación de Gawk

Última versión comprobada: 3.1.0.

(Todavía no se han comprobado las dependencias.)

Instalación de Gawk

Prepara Gawk para su compilación:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    LDFLAGS="-static" ./configure --prefix=$LFS/static --disable-nls
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de GCC–3.2.1

```
Estimación del tiempo de construcción:       9.48 SBU
Estimación del espacio necesario en disco:  326 MB
```

Contenido de GCC

Última versión comprobada: 3.1.

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

GCC instala lo siguiente:

Programas

c++, c++filt, cc (enlace a gcc), cc1, cc1plus, collect2, cpp, cpp0, g++, gcc, gccbug, gcov y tradcpp0

Librerías

libgcc.a, libgcc_eh.a, libgcc_s.so, libiberty.a, libstdc++.[a,so], libsupc++.a

Dependencias de instalación de GCC

Última versión comprobada: 2.95.3.

Bash: sh
Binutils: ar, as, ld, nm, ranlib
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch
Find: find
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, true, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tail, tr

Instalación de GCC

No necesitamos un compilador C++ hasta el Capítulo 6. Así que, en este momento, sólo es necesario desempaquetar el paquete gcc−core.

Se sabe que este programa se comporta mal si cambias sus parámetros de optimización (incluyendo las opciones −march y −mcpu). Por esta razón, si tienes definida cualquier variable de entorno que pueda sobreescribir las optimizaciones por defecto, como CFLAGS o CXXFLAGS, recomendamos quitarlas o modificarlas cuando construyas GCC.

La documentación sobre la instalación de GCG recomienda construir GCC en un directorio dedicado fuera del árbol de las fuentes:

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

Prepara GCC para su compilación:

```
../gcc-3.2.1/configure --prefix=/static \
   --disable-nls --disable-shared \
   --with-as=$LFS/static/bin/as \
   --with-ld=$LFS/static/bin/ld
```

El significado de las opciones de configure es:

- **`--prefix=/static:`** Esto NO es un error tipográfico. GCC incorpora a su código algunas rutas durante la compilación y por eso necesitamos indicar `/static` como prefijo durante la fase de configure. Pasaremos el prefijo real de instalación (`$LFS/static`) posteriormente, durante la fase de instalación.
- **`--disable-shared:`** Esto evita la construcción de las librerías dinámicas. No son de utilidad para nosotros en este momento. Las crearemos cuando reinstalemos GCC en el capítulo 6.
- **`--with-as=$LFS/static/bin/as y --with-ld=$LFS/static/bin/ld:`** GCC puede compilarse mal si el paquete Binutils de tu sistema anfitrión es algo antiguo. Necesitamos un GCC estático que funcione bien hasta que reistalemos posteriormente GCC en el capítulo 6. Por eso usamos los programas `as` y `ld` del paquete Binutils que acabamos de compilar, para asegurarnos de que GCC funcionará correctamente.

Continúa compilando el paquete:

```
make BOOT_LDFLAGS="-static"
bootstrap
```

El significado de las opciones de make es:

- **`BOOT_LDFLAGS="-static":`** Esto es el equivalente al "make LDFLAGS=–static" que usamos con otros paquetes para compilarlos estáticamente.
- **`bootstrap:`** El objetivo *bootstrap* no sólo compila GCC, si no que compila GCC varias veces. Utiliza el primer programa compilado para compilarse a si mismo una segunda y tercera vez, asegurandose de que el compilador se compiló correctamente.

Y termina instalando el paquete:

```
make prefix=$LFS/static
install-no-fixedincludes
```

El significado de la opción de make es:

- **`install-no-fixedincludes:`** Esto evita que se ejecute el guión fixincludes. Se necesita porque, bajo circunstancias normales, la instalación de GCC ejecutará el guión fixincludes que explora tu sistema en búsqueda de ficheros de cabecera que necesiten ser fijados. Digamos que encuentra ficheros de cabecera de Glibc. Las corregirá y acabarán en `$LFS/static/lib/gcc-lib/i686-pc-linux-gnu/3.2`. Más tarde, en el capítulo 6, instalarás Glibc que instalará los ficheros de cabecera en `/usr/include`. A continuación instalarás otros programas que usarán las cabeceras de Glibc y GCC las buscará en `/static/lib/gcc-lib` antes de mirar en `/usr/include`, con el resultado de que encontrará y utilizará los ficheros de cabecera de Glibc corregidos provenientes de tu sistema anfitrión, que posiblemente sean incompatibles con la versión de Glibc usada actualmente en el sistema LFS.

Como toque final crearemos el enlace simbólico `$LFS/static/bin/cc`. Muchos programas y guiones intentan ejecutar **`cc`** en lugar de **`gcc`**. Esto es así para hacer que los programas sean genéricos y utilizables en todo tipo de sistemas Unix. No todo el mundo tiene GNU CC instalado. La simple ejecución de **`cc`** (Compilador de C) deja al usuario libre para decidir el compilador de C a instalar. El enlace simbólico apuntará al compilador por defecto del sistema.

```
ln -s gcc
$LFS/static/bin/cc
```

Instalación de Grep–2.5

```
Estimación del tiempo de construcción:       0.26 SBU
Estimación del espacio necesario en disco:  5 MB
```

Contenido de Grep

Última versión comprobada: 2.5.

Grep es un programa usado para imprimir las líneas de un fichero que cumplan un patrón especificado.

Grep instala lo siguiente:

Programas

egrep (enlace a grep), fgrep (enlace a grep) y grep

Dependencias de instalación de Grep

Última versión comprobada: 2.4.2.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install–info, makeinfo
Textutils: cat, tr

Instalación de Grep

Prepara Grep para su compilación:

```
LDFLAGS="-static"
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    ./configure --prefix=$LFS/static --disable-nls \
    --disable-perl-regexp
```

El significado de la opción de configure es:

- **`--disable-perl-regexp:`** Esta opción de configure asegura que `grep` no se enlaza con la librería PCRE, que frecuentemente sólo se encuentra disponible en las distribuciones como librería compartida. No usar esta opción puede causar errores de compilación.

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Gzip–1.2.4a

```
Estimación del tiempo de construcción:       0.04 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Gzip

Última versión comprobada: 1.2.4a.

El paquete Gzip contiene programas para comprimir y descomprimir ficheros usando el codificador Lempel–Ziv (LZ77).

Gzip instala lo Siguiente:

Programas

gunzip (enlace a gzip), gzexe, gzip, uncompress (enlace a gunzip), zcat (enlace a gzip), zcmp, zdiff, zforce, zgrep, zmore y znew

Dependencias de instalación de Gzip

Última versión comprobada: 1.2.4a.

Bash: sh
Binutils: as, ld, nm

Fileutils: chmod, cp, install, ln, mv, rm
Gcc: cc1, collect2, cpp, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: hostname
Textutils: cat, tr

Instalación de Gzip

Prepara Gzip para su compilación:

```
./configure --prefix=$LFS/static
```

Continúa compilando el paquete:

```
make LDFLAGS="-static"
```

Y termina instalando el paquete:

```
make install
```

Instalación de Make−3.80

```
Estimación del tiempo de construcción:       0.26 SBU
Estimación del espacio necesario en disco:  8 MB
```

Contenido de Make

Última versión comprobada: 3.79.1.

Make determina, automáticamente, qué piezas de un programa largo es necesario recompilar y ejecuta los comandos para recompilarlas.

Make instala lo siguiente:

Programas

make

Dependencias de instalación de Make

Última versión comprobada: 3.79.1.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chgrp, chmod, install, ls, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Make

Prepara Make para su compilación:

```
LDFLAGS="-static" ./configure --prefix=$LFS/static
--disable-nls
```

Continúa compìlando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Durante la etapa make install verás este aviso:

```
chgrp: changing group of `/mnt/lfs/static/bin/make':
Operation not permitted
/mnt/lfs/static/bin/make needs to be owned by group kmem and setgid;
otherwise the `-l' option will probably not work. You may need special
privileges to complete the installation of /mnt/lfs/static/bin/make.

chgrp: cambiando grupo de `/mnt/lfs/static/bin/make': Operación no permitida
/mnt/lfs/static/bin/make necesita pertenecer al grupo kmem y activar el bit
SGID; de otra manera, la opción '-l' probablemente no funcionará. Puede
que necesites privilegios especiales para completar la instalación de
/mnt/lfs/static/bin/make.
```

Puedes ignorar este aviso sin que conlleve ninguna consecuencia. El programa **make** no necesita pertenecer al grupo kmem ni ejecutarse con ese ID de grupo (setgid) para que la opción *−l* funcione. (Esta opción le ordena

a **make** que no empiece nuevos trabajos cuando se alcance una determinada carga del sistema).

Instalación de Patch–2.5.4

```
Estimación del tiempo de construcción:       0.10 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido dePatch

Última versión comprobada: 2.5.4.

El programa patch modifica un fichero basandose en un parche. Normalmente un parche es una lista, creada por el programa diff, que contiene instrucciones sobre cómo debe modificarse el fichero original.

Patch instala lo siguiente:

Programas

patch

Dependencias de instalación de Patch

Última versión comprobada: 2.5.4.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, grep
Make: make
Sed: sed
Sh–utils: echo, expr, hostname, uname
Textutils: cat, tr

Instalación de Patch

Prepara Patch para su compilación:

```
CPPFLAGS=-D_GNU_SOURCE \
    LDFLAGS="-static" ./configure
--prefix=$LFS/static
```

El significado de la opción de configure es:

- **CPPFLAGS=-D_GNU_SOURCE:** Este modificador corrige los problemas de instalación de este paquete en las plataformas PPC y m68k (que nosotros sepamos). Tampoco daña la compilación en otras plataformas, como x86, así que lo hacemos por defecto.

Continúa instalando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Sed–4.0.5

```
Estimación del tiempo de construcción:       0.09 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Sed

Última versión comprobada: 3.02.

sed es un editor de flujo. Un editor de flujo se utiliza para realizar transformaciones básicas de texto sobre un flujo de entrada (un fichero o la entrada procedente de una tubería).

Sed instala lo siguiente:

Programas

sed

Dependencias de instalación de Sed

Última versión comprobada: 3.02.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh–utils: echo, expr, hostname, sleep

Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Sed

Prepara Sed para su compilación:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    LDFLAGS="-static" ./configure --prefix=$LFS/static --disable-nls
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Sh−utils−2.0

```
Estimación del tiempo de construcción:       0.47 SBU
Estimación del espacio necesario en disco:  42 MB
```

Contenido de Sh−utils

Última versión comprobada: 2.0.

El paquete Sh−utils contiene un número de utilidades para realizar manipulaciones básicas en el intérprete de comandos.

Sh−utils instala lo siguiente:

Programas

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami y yes

Dependencias de instalación de Sh−utils

Última versión comprobada: 2.0.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh

Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, chown, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Sh−utils

Antes de instalar Sh−utils, puede ser necesario aplicarle un parche para evitar un conflicto de nombres de variables con ciertas versiones de Glibc (normalmente glibc−2.1.x) cuando se compila sh−utils estáticamente. De todas formas, es inocuo aplicar el parche incluso si tienes una versión diferente de glibc. De modo que, si no estás seguro, es mejor que lo apliques.

```
patch -Np1 -i
../sh-utils-2.0.patch
```

Prepara Sh−utils para su compilación:

```
LDFLAGS="-static" ./configure --prefix=$LFS/static \
    --disable-nls
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Cuando ejecutes make install verás el siguiente mensaje de aviso:

```
WARNING: insufficient access; not installing su
NOTE: to install su, run 'make install-root' as root

AVISO: privilegios insuficientes; no se instalará su
NOTA: para instalar su, ejecuta 'make install-root' como usuario
root
```

Puedes ignorar este mensaje de aviso. Necesitas entrar al sistema como root para instalar `su` de la manera que Sh−utils quiere hacerlo, es decir, activando el bit SUID. No necesitamos `su` durante el Capítulo 6 y se instalará correctamente cuando reinstalemos Sh−utils en dicho capítulo.

Instalación de Tar−1.13

```
Estimación del tiempo de construcción:       0.25 SBU
Estimación del espacio necesario en disco:  10 MB
```

Contenido de Tar

Última versión comprobada: 1.13.

Tar es un programa de archivado diseñado para almacenar y estraer ficheros en un archivo conocido como fichero tar.

Tar instala lo siguiente:

Programas

rmt y tar

Dependencias de instalación de Tar

Última versión comprobada: 1.13.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Patch: patch
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Tar

Debe aplicarse un parche para darle a tar soporte directo de ficheros bzip2. Este parche añade la opción *–j* a tar, que es similar a la opción *–z* usada para los ficheros gzip.

Aplica el parche ejecutando el siguiente comando:

```
patch -Np1 -i ../tar-1.13.patch
```

Prepara Tar para su compilación:

```
LDFLAGS="-static" ./configure --prefix=$LFS/static --disable-nls
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Texinfo–4.3

```
Estimación del tiempo de construcción:       0.47 SBU
Estimación del espacio necesario en disco:  19 MB
```

Contenido de Texinfo

Última versión comprobada: 4.2.

El paquete Texinfo contiene programas usados para leer, escribir y convertir documentos Info, que suministran documentación del sistema.

Texinfo instala lo siguiente:

Programas

info, infokey, install–info, makeinfo, texi2dvi y texindex

Dependencias de instalación de Texinfo

Última versión comprobada: 4.0.

Bash: sh
Binutils: ar, as, ld, ranlib

Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep
Texinfo: makeinfo
Textutils: cat, tr

Instalación de Texinfo

Prepara Texinfo para su compilación:

```
LDFLAGS="-static" ./configure --prefix=$LFS/static \
    --disable-nls
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Textutils−2.1

```
Estimación del tiempo de construcción:       0.95 SBU
Estimación del espacio necesario en disco:  49 MB
```

Contenido deTextutils

Última versión comprobada: 2.0.

El paquete Textutils contiene varios programas para procesar y manipular ficheros de texto.

Textutils instala lo siguiente:

Programas

cat, cksum, comm, csplit, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq y wc

Dependencias de instalación de Textutils

Última versión comprobada: 2.0.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Textutils

Prepara Textutils para su compilación:

```
CPPFLAGS=-Dre_max_failures=re_max_failures2 \
    LDFLAGS="-static" ./configure --prefix=$LFS/static \
    --disable-nls
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Util−linux−2.11y

```
Estimación del tiempo de construcción:       0.09 SBU
Estimación del espacio necesario en disco:  9 MB
```

Contenido de Util−linux

Última versión comprobada: 2.11t.

El paquete Util−linux contiene una miscelánea de utilidades. Alguna de estas utilidades más destacables se utilizan para montar, desmontar, formatear, particionar y manejar dispositivos de disco, abrir puertos de consola o capturar los mensajes del núcleo.

Util−linux instala lo siguiente:

Programas

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, 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, parse.bash, parse.tcsh, pg, pivot_root, ramsize (enlace a rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (enlace a rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (enlace a swapon), swapon, test.bash, test.tcsh, tunelp, ul, umount, vidmode (enlace a rdev), whereis y write

Dependencias de instalación de Util−linux

Última versión comprobada: 2.11n.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chgrp, chmod, cp, install, ln, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp, cpp0
Glibc: rpcgen
Grep: grep
Make: make
Sed: sed
Sh−utils: uname, whoami
Textutils: cat

Instación de Util−linux

Prepara Util−linux para su compilación:

```
LDFLAGS="-static" ./configure
```

Continúa compilando el paquete:

```
make -C lib
```

Por ahora solo necesitamos los programas *mount* y *umount*, por lo que no compilaremos el paquete completo. Compila estos dos programas ejecutando el siguiente comando:

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

Y termina instalando estos dos programas:

```
cp mount/{mount,umount} $LFS/static/bin
```

III. Parte III – Construcción del sistema LFS

Índice

6. Instalación de los programas del sistema base
7. Preparando los guiones de arranque
8. Hacer el sistema LFS arrancable
9. El final

Capítulo 6. Instalación de los programas del sistema base

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, creamos algunas cosas auxiliares y, después, comenzamos a instalar todos los paquetes uno por uno.

La instalación de todos estos paquetes es algo bastante sencillo, por lo que puedes pensar que, probablemente, sea más corto dar aquí las instrucciones genéricas de instalación y sólo explicar en profundidad la instalación de los paquetes que necesiten un método alternativo. Aunque estemos de acuerdo en eso, hemos elegido dar las instrucciones completas para todos y cada uno de los paquetes, simplemente para minimizar la posibilidad de errores.

Si piensas usar optimizaciones para la compilación durante este capítulo, mírate la receta de optimización en http://www.escomposlinux.org/lfs−es/recetas/optimization.html (el original se encuentra en http://hints.linuxfromscratch.org/hints/optimization.txt). Optimizar la compilación puede hacer que un programa funcione rápido, pero también puede causar problemas de compilación. Si un paquete rehúsa compilar usando optimización, inténtalo sin optimización para ver si el problema persiste.

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 `/static`. 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 `/static`, lo que provocaría que el programa dejase de funcionar cuando se elimine dicho directorio.

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).

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.

Pero 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 de qué compilador se usó y con qué librería C. 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 eliminar los símbolos de depuración de un binario (que debe ser un binario a.out o ELF) ejecuta **strip --strip-debug fichero**. Pueden usarse comodines para procesar múltiples ficheros (utilizando algo como: **strip --strip-debug $LFS/usr/bin/***).

Para tu comodidad, en el Capítulo 9 se incluye un comando simple para eliminar todos los símbolos de depuración de los programas y librerías del sistema. Puedes encontrar información adicional en la receta de optimización que hay en http://www.escomposlinux.org/lfs-es/recetas/optimization.html (el original se encuentra en http://hints.linuxfromscratch.org/hints/optimization.txt).

Entrando al entorno chroot

Es la hora de entrar en el entorno chroot para instalar los paquetes que necesitamos. Antes de que puedas hacer chroot, sin embargo, necesitas cambiar al usuario *root*, pues sólo el usuario *root* puede usar el comando **chroot**.

Hazte *root* y ejecuta el siguiente comando para entrar al entorno chroot:

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

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 fijará la variable TERM dentro del chroot al mismo valor que fuera del chroot, pues programas como vim y less la necesitan para funcionar correctamente. Si necesitas tener presentes otras variables, como CFLAGS o CXXFLAGS, éste 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 -- ya que lo que el intérprete de comandos piensa que es `/` en realidad es el valor de `$LFS`, que se le pasó al comando chroot.

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), debes recordar entrar en el chroot y montar proc (como explicaremos más tarde) antes de seguir con las instalaciones.

Ten en cuenta que en la línea de entrada de comandos de bash pondrá: "I have no name!". Esto es normal porque Glibc no ha sido instalada todavía.

Cambio del propietario

En estos momentos el directorio /static pertenece al usuario lfs. Sin embargo, esta cuenta de usuario sólo existe en el sistema anfitrión. Aunque puedes borrar el directorio `/static` una vez que hayas terminado tu sistema LFS, puede que quieras conservarlo para, por ejemplo, construir más sistemas LFS. Pero si guardas el directorio `/static` acabarás con ficheros que pertenecen a un identificador de usuario sin su cuenta correspondiente. Esto es peligroso porque una cuenta de usuario creada posteriomente podría tener esta

identidad de usuario y poseería repentinamente los derechos sobre el directorio `/static` y todos los ficheros que contiene. Esto abriría el directorio `/static` a la manipulación de un usuario que no es de confianza.

Para evitar este problema, puedes añadir posteriormente el usuario *lfs* al nuevo sistema LFS, cuando creemos el ficero `/etc/passwd`, teniendo cuidado de asignarle los mismos identificadores de usuario y grupo. Alternativamente, puedes (y el libro asume que lo haces) asignar el contenido del directorio `/static` al usuario *root* ejecutando el siguiente comando:

```
chown -R 0:0 /static
```

Este comando utiliza "0:0" en lugar de "root:root", pues chown no es capaz de resolver el nombre "root" hasta que glibc sea instalada.

Creación de los directorios

Ahora vamos a crear una estructura en nuestro sistema de ficheros LFS. Crearemos un árbol de directorios. Usando los siguientes comandos se creará un árbol más o menos estándar:

```
mkdir -p /{bin,boot,dev/pts,etc/opt,home,lib,mnt,proc} &&
mkdir -p /{root,sbin,tmp,usr/local,var,opt} &&
for dirname in /usr /usr/local
    do
    mkdir $dirname/{bin,etc,include,lib,sbin,share,src}
    ln -s share/{man,doc,info} $dirname
    mkdir $dirname/share/{dict,doc,info,locale,man}
    mkdir $dirname/share/{nls,misc,terminfo,zoneinfo}
    mkdir $dirname/share/man/man{1,2,3,4,5,6,7,8}
done &&
mkdir /usr/lib/locale &&
mkdir /var/{lock,log,mail,run,spool} &&
mkdir -p /var/{tmp,opt,cache,lib/misc,local} &&
mkdir /opt/{bin,doc,include,info} &&
mkdir -p /opt/{lib,man/man{1,2,3,4,5,6,7,8}} &&
ln -s ../var/tmp /usr
```

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

```
chmod 0750 /root &&
chmod 1777 /tmp /var/tmp
```

El primer cambio nos asegura que no todo el mundo 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 prohíbe el llamado "bit pegajoso" (sticky bit) – el bit más alto de la máscara de permisos 1777.

Nota de conformidad con FHS

Basamos nuestro árbol de directorios en el estándar FHS (disponible en http://www.pathname.com/fhs/). Además del árbol arriba creado, este estándar estipula la existencia de `/usr/local/games` y `/usr/share/games`, pero no nos gustan para un sistema base. Sin embargo, eres libre de hacer que tu

sistema cumpla el FHS. Como sobre la estructura del subdirectorio `/usr/local/share` el FHS no hace precisiones, creamos aquí los directorios que pensamos que son necesarios.

Montando el sistema de ficheros /proc

Para funcionar correctamente, algunos programas necesitan que el sistema de ficheros proc esté disponible dentro del entorno chroot. Como un sistema de ficheros se puede montar tantas veces y en tantos lugares como quieras, no es problema que el sistema de ficheros proc esté todavía montado en tu sistema anfitrión −− sobre todo porque proc es un sistema de ficheros virtual.

El sistema de ficheros proc se monta en `/proc` ejecutando el siguiente comando.

```
mount proc /proc -t proc
```

Posiblemente el comando mount te muestre algunos mensajes de aviso como este:

```
warning: can't open /etc/fstab: No such file or directory
not enough memory

aviso: no se puede abrir /etc/fstab: No existe el fichero o directorio
memoria insuficiente
```

Ignóralo. Está causado por el hecho de que el sistema aún no se ha instalado por completo y faltan algunos ficheros. El montaje se realizará correctamente, que es todo lo que necesitamos en este momento.

El último error (not enough memory, memoria insuficiente) no se muestra siempre. Depende de la configuración de tu sistema (como la versión de Glibc de tu sistema anfitrión usada para compilar el programa mount).

Creación del fichero mtab

Lo siguiente por hacer es crear el fichero `/etc/mtab`. Esto se hace con el siguiente comando:

```
touch /etc/mtab
```

Desde la versión 1.11 de los guiones de arranque de LFS, este fichero es manejado correctamente para que no haya errores aún al reiniciar por un cuelgue del sistema.

Creación de los enlaces simbólicos bash y sh

Algunos programas tienen fijadas en su código rutas a programas que aún no existen. Para satisfacer a estos programas creamos los enlaces simbólicos `/bin/bash` y `/bin/sh`, ambos apuntando al programa estático `bash`.

Crea los enlaces simbólicos `/bin/bash` y `/bin/sh` ejecutando los siguientes comandos:

```
ln -s /static/bin/bash /bin/bash &&
ln -s bash /bin/sh
```

Creación de los ficheros de contraseñas y grupos

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:

```
echo "root:x:0:0:root:/root:/bin/bash" >
/etc/passwd
```

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:
EOF
```

Los grupos creados no son parte de ningún estándar –– son los grupos que el guión MAKEDEV utiliza en la siguiente sección. Aparte del grupo "root", el LSB (http://www.linuxbase.org) solamente recomienda que 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 paquetes correctamente escritos no dependen del número GID, sino que utilizan el nombre del grupo.

Creación de los dispositivos (Makedev–1.7)

```
Estimación del tiempo de construcción:       0.07 SBU
Estimación del espacio necesario en disco:  50 KB
```

Contenido de MAKEDEV

Última versión comprobada: 1.5.

MAKEDEV es un guión que crea los ficheros de dispositivos estáticos necesarios, que usualmente residen en el directorio `/dev`. Puede encontrarse más información sobre los ficheros de dispositivos dentro de las fuentes del núcleo en `Documentation/devices.txt`.

MAKEDEV instala lo siguiente:

Programas

MAKEDEV

Dependencias de instalación de MAKEDEV

Última versión comprobada: 1.5.

Bash: sh
Fileutils: chmod, chown, cp, ln, mknod, mv, rm
Grep: grep
Sh−utils: expr, id

Creación de los dispositivos

Ten en cuenta que al dempaquetar el fichero MAKEDEV−1.7.bz2 no se crea un directorio al que debas entrar, pues el fichero sólo contiene un guión del intérprete de comandos.

Instala el guión MAKEDEV:

```
cp MAKEDEV-1.7
/dev/MAKEDEV
```

Prepara el guión para su ejecución:

```
cd /dev &&
chmod 754 MAKEDEV
```

Ejecuta el guión para crear los ficheros de dispositivos:

```
./MAKEDEV -v generic
```

Si no encuentras un dispositivo que necesitas, prueba a ejecutar **`./MAKEDEV -v <device>`**. Alternativamente, puedes crear los dispositivos mediante el programa *mknod*. Consulta las páginas de manual e info de *mknod* si necesitas más información.

Si piensas utilizar *devpts*, pásale la opción **`generic-nopty`** a `MAKEDEV`. Esto creará los mismos dispositivos que **`generic`**, pero se saltará la creación de los dispositivos *pty*, pues *devpts* puede manejarlos por si solo.

Instalación de las cabeceras de Linux−2.4.20

```
Estimación del tiempo de construcción:       0.02 SBU
Estimación del espacio necesario en disco:  142 MB
```

Contenido de Linux

Última versión comprobada: 2.4.18.

El núcleo Linux es el corazón de todo sistema Linux. Es lo que hace a Linux funcionar. Cuando se enciende un ordenador y se inicia un sistema Linux, el núcleo es lo primero que se carga. El núcleo inicializa los componentes hardware del sistema: puertos serie, puertos paralelo, tarjetas de sonido, tarjetas de red, controladores IDE, controladores SCSI y mucho más. En pocas palabras, el núcleo hace que el hardware esté disponible para que el software pueda ejecutarse.

Linux instala lo siguiente:

Programas

El núcleo y las cabeceras del núcleo

Dependencias de instalación de Linux

Última versión comprobada: 2.4.17.

Bash: sh
Binutils: ar, as, ld, nm, objcopy
Fileutils: cp, ln, mkdir, mv, rm, touch
Findutils: find, xargs
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Gzip: gzip
Make: make
Gawk: awk
Modutils: depmod, genksyms
Net−tools: dnsdomainname, hostname
Sed: sed
Sh−utils: basename, date, expr, pwd, stty, uname, whoami, yes
Textutils: cat, md5sum, sort, tail

Instalación de las cabeceras del núcleo

No compilaremos todavía un nuevo núcleo (kernel) −− lo haremos cuando terminemos la instalación de todos los paquetes. Pero como ciertos paquetes necesitan los ficheros de cabecera (headers) del núcleo, vamos a desempaquetar el archivo del núcleo ahora, ajustarlo, y copiar los ficheros de cabecera a un lugar donde puedan encontrarlos esos paquetes.

Es importante notar que los ficheros del directorio de las fuentes del núcleo no pertenecen a *root*. Aunque desempaquetes un paquete como usuario *root* (como hacemos dentro del chroot), los ficheros acaban teniendo

los identificadores de usuario y grupo que tenían en la computadora en la que se empaquetó. Esto usualmente no es un problema para otros paquetes que instales debido a que eliminas el árbol de las fuentes despues de la instalación. Pero el árbol de las fuentes del núcleo Linux se guarda con frecuencia durante mucho tiempo, por lo que si en alguna ocasión ese identificador es asignado a alguien en tu máquina, esa persona obtendrá permisos de escritura sobre las fuentes del núcleo.

A la vista de esto, puede que quieras ejecutar **`chown -R 0:0`** en el directorio `linux-2.4.20` para asegurarte de que todos los ficheros son propiedad del usuario *root*.

La instalación de las cabeceras del núcleo requiere del programa *pwd*. En las fuentes del núcleo, la ruta al programa pwd está fijada como /bin/pwd. Crea un enlace simbólico para solventar esto:

```
ln -s /static/bin/pwd
/bin/pwd
```

Prepara la instalación de las cabeceras:

```
make mrproper
```

Esto asegurará que el árbol del núcleo está absolutamente limpio. El equipo de desarrollo del núcleo recomienda usar este comando antes de *cada* compilación del núcleo, y en realidad no debes confiar en que el árbol de las fuentes esté limpio tras desempaquetarlo.

Crea el fichero `include/linux/version.h`:

```
make
include/linux/version.h
```

Crea el enlace simbólico `include/asm` específico de la plataforma:

```
make symlinks
```

Instala los ficheros de cabecera específicos de la plataforma:

```
cp -HR include/asm /usr/include &&
cp -R include/asm-generic /usr/include
```

Instala los ficheros de cabecera del núcleo independientes de la plataforma:

```
cp -R include/linux
/usr/include
```

Hay ciertos fichero de cabecera del núcleo que hacen uso del fichero de cabecera `autoconf.h`. Puesto que todavía no hemos configurado el núcleo, necesitamos crear este fichero por nuestra cuenta para evitar fallos de compilación. Crea un fichero `autoconf.h` vacio:

```
touch
/usr/include/linux/autoconf.h
```

Como el enlace simbólico `/bin/pwd` que creamos anteriormente era sólo temporal, podemos eliminarlo ahora:

```
rm /bin/pwd
```

Por qué copiamos las cabeceras del núcleo y no las enlazamos simbólicamente.

En el pasado, era una práctica común enlazar simbólicamente los directorios `/usr/include/{linux,asm}` a `/usr/src/linux/include/{linux,asm}`. Esta fue una *mala* práctica, como señala este extracto de un mensaje de Linus Torvalds a la Lista de Correo del Núcleo Linux:

```
Sugeriría que la gente que compile núcleos nuevos debe:

 - no tener un sólo enlace simbólico a la vista (excepto el que crea la
   misma construcción del núcleo, el enlace simbólico llamado
   "linux/include/asm", que sólo se usa para la compilación interna del
   mismo núcleo).

Y sí, esto es lo que yo hago. Mi /usr/src/linux todavía contiene los
ficheros de cabecera del antiguo 2.2.13, aunque no he ejecutado un
núcleo 2.2.13 desde hace _muucho_ tiempo. Pero esas fueron las cabeceras
con las que fue compilada glibc, por lo que esas cabeceras son las que
coinciden con los ficheros objeto de la librería.

Y este es, de hecho, el entorno que se ha sugerido en, al menos, los
últimos cinco años. No sé por qué el asunto del enlace simbólico sigue
coleando, como un mal zombi. Casi cada distribución todavía tiene ese
enlace simbólico roto,  y la gente todavía recuerda que el código fuente
de linux debe ir en "/usr/src/linux" aunque no ha sido cierto desde hace
_muucho_ tiempo.
```

La parte relevante es donde Linus afirma que las cabeceras deberían ser con las que *fue compilada glibc*. Estas son las cabeceras que deberías usar cuando más adelante compiles otros paquetes, pues son las que coinciden con los códigos−objeto de las librerías. Al copiar las cabeceras nos aseguramos de que permanecen disponibles si posteriormente actualizas el núcleo.

Advierte, de paso, que es perfectamente correcto tener las fuentes del núcleo en `/usr/src/linux`, mientras no tengas los enlaces simbólicos `/usr/include/{linux,asm}`.

Instalación de Man−pages−1.54

```
Estimación del tiempo de construcción:       0.01 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Man−pages

Última versión comprobada: 1.54.

El paquete Man−pages contiene alrededor de 1200 páginas de manual. Esta documentación detalla las funciones de C y C++, describe varios ficheros de dispositivo importantes y proporciona documentación procedente de otros paquetes que posiblemente falte en los mismos.

Man−pages instala lo siguiente:

Ficheros de soporte

Varias páginas de manual.

Dependencias de instalación de Man−pages

Última versión comprobada: 1.47.

Bash: sh
Fileutils: install
Make: make

Instalación de Man−pages

Instala Man−pages ejecutando el siguiente comando:

```
make install
```

Instalación de Glibc−2.3.1

```
Estimación del tiempo de construcción:       14.71 SBU
Estimación del espacio necesario en disco:  369 MB
```

Contenido de Glibc

Última versión comprobada: 2.2.5.

Glibc es la librería C que proporciona las llamadas al sistema y las funciones básicas, tales como open, malloc, printf, etc. La librería C es utilizada por todos los programas enlazados dinámicamente.

Glibc instala lo siguiente:

Programas

catchsegv, gencat, getconf, getent, glibcbug, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, nscd_nischeck, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump y zic

Librerías

ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd−compat.a, libc.[a,so], libc_nonshared.a, libcrypt.[a,so], libdl.[a,so], libg.a, libieee.a, libm.[a,so], libmcheck.a, libmemusage.so, libnsl.a, 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]

Dependencias de instalación de Glibc

Última versión comprobada: 2.2.5.

Bash: sh
Binutils: ar, as, ld, ranlib, readelf
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mknod, mv, mkdir, rm, touch
Gcc: cc, cc1, collect2, cpp, gcc
Grep: egrep, grep
Gzip: gzip
Make: make
Gawk: gawk
Sed: sed
Sh−utils: date, expr, hostname, pwd, uname
Texinfo: install−info, makeinfo
Textutils: cat, cut, sort, tr

Instalación de Glibc

Antes de instalar glibc, debes entrar al directorio `glibc-2.3.1` usando el comando **`cd`** y desempaquetar Glibc−linuxthreads dentro de este directorio, no en /usr/src como normalmente harías.

Se sabe que este programa se comporta mal si cambias sus parámetros de optimización (incluyendo las opciones −march y −mcpu). Por tanto, si tienes definida cualquier variable de entorno que pueda sobreescribir las optimizaciones por defecto, como CFLAGS y CXXFLAGS, te recomendamos que las desactives antes de construir Glibc.

Básicamente, compilar Glibc de forma diferente a como el libro sugiere pone tu sistema en grave riesgo.

Comenzaremos aplicando un parche que hace lo siguiente:

- Convierte todas las apariciones de *$(PERL)* a */usr/bin/perl* en el fichero `malloc/Makefile`. Se hace esto porque Glibc no puede autodetectar la localización de `perl`, pues el paquete Perl no se ha instalado todavía. Y si Glibc piensa que Perl no está instalado, el programa perl `mtrace` no será instalado.
- Sustituye todas la apariciones de *root* con *0* en el fichero `login/Makefile`. Esto se hace porque la propia Glibc no se ha instalado aún y, por tanto, todavía no funciona la resolución de los nombres de usuario a su identificador, así que un **`chown root fichero`** fallará. Si usamos el identificador numérico (como **`chown 0 fichero`**) funciona perfectamente.

```
patch -Np1 -i ../glibc-2.3.1-root-perl.patch
```

Hay un problema potencial que causa que fallen los binarios enlazados estáticamente contra una librería Glibc−2.2 o anterior. Aunque los binarios estáticos tienen incorporadas todas las partes necesarias de Glibc,

todavía se apoyan en un conjunto de librerías externas: las librerías NSS de Glibc. Estas librerías, entre otras cosas, indican a los programas dónde se encuentra la base de datos con las contraseñas del sistema (en `/etc/password`, NIS, o cualquier otro sistema que se haya configurado).

Glibc ha sufrido algunos cambios desde la versión 2.2.x y el nuevo código NSS es incompatible con el antiguo. Así que cuando se instale Glibc se instalarán las nuevas librerías NSS, los programas estáticos cargarán estas nuevas librerías NSS y abortarán con *segmentation fault* (violación de segmento). Este parche deshace algunos de los cambios para evitar el problema.

Si construiste los programas del capítulo 5 en un sistema anfitrión que utilizase Glibc−2.2.x o anterior, debes aplicar el siguiente parche. Instalaremos Glibc otra vez al final de este capítulo para eliminar este parche, con lo que tendrás una Glibc púra tal y como sugieren los desarrolladores.

```
patch -Np1 -i ../glibc-2.3.1-libnss.patch
```

Glibc comprobará el fichero `/etc/ld.so.conf` y abortará con un error si no lo encuentra, así que debemos crearlo:

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

La documentación de instalación de Glibc recomienda construir Glibc en un directorio aparte, y no en el directorio de las fuentes:

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

A continuación, prepara Glibc para su compilación:

```
../glibc-2.3.1/configure --prefix=/usr \
    --disable-profile --enable-add-ons \
    --libexecdir=/usr/bin
```

El significado de las opciones de configure es:

- **`--disable-profile:`** Esto desactiva la construcción de librerías con información de perfiles. Este comando puede omitirse si planeas usar perfiles.
- **`--enable-add-ons:`** Esto activa el añadido que se instalará con Glibc, en nuestro caso linuxthreads
- **`--libexecdir=/usr/bin:`** Esto hará que el programa `pt_chown` se instale en el directorio `/usr/bin`.

Durante la fase de configuración verás los siguientes mensajes de aviso:

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

configure: aviso:
*** Este programa auxiliar no se ha encontrado o es demasiado antiguo: msgfmt
*** algunas características serán desactivadas.
*** Compruebe en el fichero INSTALL la versión requerida.
```

Que no se encuentre `msgfmt` (incluido en el paquete Gettext que será instalado más adelante en este capítulo) no causa ningún problema. Se usa `msgfmt` para generar los ficheros binarios traducidos que se usan para hacer que el sistema "hable" en un idioma diferente. Como estos ficheros de traducción ya han sido generados para ti, no es necesario `msgfmt`. Solamente necesitas `msgfmt` si cambias los ficheros fuente de traducción (los ficheros `*.po` del subdirectorio `po`) lo cual requerirá regenerar los ficheros binarios.

Debido a que Glibc aún no ha sido instalada, una de las pruebas lanzadas por el guión configure fallará. Esta prueba se supone que comprueba `gcc` para determinar si hay o no instalado un compilador para plataforma cruzada. Sin embargo, hace falte tener instalada Glibc para lanzar esta prueba. Puesto que la prueba falla, el guión configure asume automáticamente que hacemos compilación cruzada. Tenemos que evitar esta asunción indicandole explícitamente a Glibc que no queremos una compilación cruzada. No hacer esto prococa efectos indeseables, como que no se instalen los ficheros de las zonas horarias.

```
echo "cross-compiling = no" > configparms
```

Continúa compilando el paquete:

```
make
```

Continuaremos instalando el paquete. Las páginas de manual de Linuxthreads no van a instalarse en este punto debido a que necesitan una instalación funcional de Perl. Instalaremos Perl más tarde en este capítulo, y las páginas de manual serán instaladas cuando se instale Glibc por segunda vez, al final de este capítulo.

```
make install
```

Las locales (utilizadas por la Glibc para hacer que el sistema "hable" en un idioma diferente) no se instalan cuando lanzas **`make install`**, por lo que tendremos que hacerlo nosotros ahora:

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

Una alternativa al comando anterior es instalar solamente aquellas locales que necesites o desees. Esto puede hacerse usando el comando localedef. Se puede encontrar más información sobre esto en el fichero `INSTALL` del árbol de `glibc-2.3.1`.

Para finalizar la instalación recargaremos Bash para que utilice los nuevos ficheros `libnss_*`. Esto también nos librará del mensaje *I have no name!* de la línea de comandos:

```
exec /static/bin/bash
--login
```

Configuración de Glibc

Necesitamos crear el fichero `/etc/nsswitch.conf`. Aunque glibc debería faciltarnos los valores por defecto cuando este fichero no se encuentra o está corrupto, estos valores por defecto no funcionan bien con la conexión de red. Esto se tratará en un capítulo posterior. También tendremos que configurar nuestra zona horaria.

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

```
cat > /etc/nsswitch.conf << "EOF"
# Inicio de /etc/nsswitch.conf

passwd: files
group: files
shadow: files

publickey: files

hosts: files dns
networks: files

protocols: db files
services: db files
ethers: db files
rpc: db files

netgroup: db files

# Fin de /etc/nsswitch.conf
EOF
```

Para saber en cuál zona horaria estás, ejecuta este guión:

```
tzselect
```

Luego de contestar unas preguntas referentes a tu localización, el guión te mostrará el nombre del fichero con tu zona horaria, algo como *EST5EDT* o *Canada/Eastern*. Crea entonces el enlace simbólico `/etc/localtime` ejecutando:

```
ln -sf ../usr/share/zoneinfo/Canada/Eastern
/etc/localtime
```

Por supuesto, reemplaza *Canada/Eastern* por el nombre de la zona horaria que te dió el guión **`tzselect`**.

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 en el 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"
# Inicio de /etc/ld.so.conf

/usr/local/lib
```

```
/opt/lib

# Fin de /etc/ld.so.conf
EOF
```

Instalación de GCC–3.2.1

```
Estimación del tiempo de construcción:       13.26 SBU
Estimación del espacio necesario en disco:  221 MB
```

Contenido de GCC

Última versión comprobada: 3.1.

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

GCC instala lo siguiente:

Programas

c++, c++filt, cc (enlace a gcc), cc1, cc1plus, collect2, cpp, cpp0, g++, gcc, gccbug, gcov y tradcpp0

Librerías

libgcc.a, libgcc_eh.a, libgcc_s.so, libiberty.a, libstdc++.[a,so], libsupc++.a

Dependencias de instalación de GCC

Última versión comprobada: 2.95.3.

Bash: sh
Binutils: ar, as, ld, nm, ranlib
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch
Find: find
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh–utils: basename, dirname, echo, expr, hostname, sleep, true, uname
Tar: tar
Texinfo: install–info, makeinfo
Textutils: cat, tail, tr

Instalación de GCC

Se sabe que este programa se comporta mal si cambias sus parámetros de optimización (incluyendo las opciones −march y −mcpu). Por esta razón, si tienes definida cualquier variable de entorno que pueda sobreescribir las optimizaciones por defecto, como CFLAGS o CXXFLAGS, recomendamos quitarlas o modificarlas cuando construyas GCC.

En este momento construiremos los compiladores C y C++, por lo que necesitarás desempaquetar tanto el paquete gcc−core como el gcc−g++. Hay otros compiladores disponibles dentro del paquete gcc completo, cuyas instrucciones de construcción puedes encontrar en http://www.escomposlinux.org/lfs−es/blfs−es−CVS/general/gcc.html (el original en inglés está en http://beyond.linuxfromscratch.org/view/cvs/general/gcc.html).

La documentación para la instalación de GCC recomienda construir GCC en un directorio fuera del árbol de las fuentes. Crea el directorio de construcción:

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

Prepara GCC para su compilación:

```
../gcc-3.2.1/configure --prefix=/usr
--enable-shared \
    --enable-threads=posix --with-slibdir=/lib \
    --enable-__cxa_atexit
--enable-clocale=gnu
```

El significado de los opciones de configure es:

- **--enable-threads=posix:** Esto activa las excepciones C++ para el manejo de código multihilo.
- **--enable-__cxa_atexit:** Esta opción dará como resultado librerías compartidas C++ y programas C++ interoperables con otras distribuciones linux.
- **--enable-clocale=gnu:** Hay un riesgo de que ciertas personas construyan librerías C++ incompatibles con ABI si no instalan todas las localedata de glibc. Utilizando −−enable−clocale=gnu nos aseguramos de que se hace "lo correcto" en todos los casos. Si no deseas usar esta opción, entonces al menos contruye la locale *de_DE*. Cuando GCC encuentra esta locale específica, entonces se implementa el modo locale correcto (*gnu*).

Continúa compilando el paquete:

```
make bootstrap
```

El objetivo *bootstrap* no sólo compila GCC, si no que compila GCC varias veces. Utiliza el primer programa compilado para compilarse a si mismo una segunda y tercera vez, asegurandose de que el compilador se compiló correctamente.

Termina instalando el paquete:

```
make
install-no-fixedincludes
```

Algunos paquetes esperan que el Pre−Procesador de C esté instalado en los directorios `/lib` y `/usr/lib`. Para satisfacer a estos paquetes, crea dos enlaces simbólicos:

```
ln −s ../usr/bin/cpp /lib &&
ln −s ../bin/cpp /usr/lib
```

Muchos paquetes compilan usando *cc* como nombre del compilador C. Para satisfacer a estos paquetes, crea el enlace simbólico *cc*:

```
ln −s gcc /usr/bin/cc
```

Instalación de Zlib−1.1.4

```
Estimación del tiempo de construcción:       0.07 SBU
Estimación del espacio necesario en disco:  1 MB
```

Contenido de Zlib

Última versión comprobada: 1.1.4.

El paquete Zlib contiene la librería zlib, utilizada por muchos programas para realizar las funciones de compresión y descompresión..

Zlib instala lo siguiente:

Librerías

libz[a,so]

Dependencias de instalación de Zlib

Dependencias no comprobadas todavía.

Instalación de Zlib

Prepara Zlib para su compilación:

```
CFLAGS="$CFLAGS −fPIC" \
    ./configure −−prefix=/usr −−shared
```

La opción *−fPIC* ayuda para asegurar la calidad de la librería dinámica zlib.

Algunos paquetes esperan que la librería zlib estática esté presente en el sistema. Para satisfacer a estos programas, compila tanto la librería compartida como la estática:

```
make LIBS="libz.so.1.1.4 libz.a"
```

Instala las librerías:

```
make LIBS="libz.so.1.1.4 libz.a" install
```

La librería compartida de zlib debe instalarse en el directorio `/lib`. De este modo, en el caso de que debas arrancar sin el directorio `/usr`, los programas vitales del sistema todavía tendrán acceso a la librería:

```
mv /usr/lib/libz.so.* /lib
```

El enlace simbólico `/usr/lib/libz.so` apunta a un fichero que no existe, debido a que lo hemos movido. Crea un enlace simbólico a la nueva localización de la librería:

```
ln -sf ../../lib/libz.so.1 /usr/lib/libz.so
```

Zlib no instala su página de manual. Ejecuta el siguiente comando para instalar esta documentación:

```
cp zlib.3 /usr/share/man/man3
```

Instalación de Findutils–4.1

```
Estimación del tiempo de construcción:       0.10 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido de Findutils

Última versión comprobada: 4.1.

El paquete Findutils contiene pogramas para encontrar ficheros, tanto al vuelo (haciendo una búsqueda recursiva en vivo a través de los directorios y mostrando sólo los ficheros que cumplan las especificaciones) o mediante una búsqueda a través de una base de datos.

Findutils instala lo siguiente:

Programas

bigram, code, find, frcode, locate, updatedb y xargs

Dependencias de instalación de Findutils

Última versión comprobada: 4.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Grep: egrep, grep

Gcc: cc1, collect2, cpp0, gcc
Make: make
Patch: patch
Sed: sed
Sh−utils: basename, date, echo, hostname
Textutils: cat, tr

Instalación de Findutils

El paquete Findutils es bastante antiguo y tiene problemas al compilarlo con versiones recientes de Glibc (Glibc−2.0.x y superiores). Necesitas aplicar el parche que soluciona estos conflictos:

- Findutils declara una función llamada basename, pero este nombre de función ya es utilizado por las librerías estandar de C. Este parche cambia la función basename de Findutils para que pase a llamarse basename2.
- Findutils usa las librerías de una forma incorrecta. Declara funciones de librería para decirle al compilador cuál será el nombre de la función cuando las librerías sean enlazadas. Dicha sintaxis es inválida, provocando advertencias y errores del compilador. Este parche elimina la sintaxis incorrecta.
- Todos los paquetes GNU deben usar una macro llamada _GNU_SOURCE. Esta macro hace que se incluyan las declaraciones de las funciones de extensión de la librería GNU para que el compilador pueda detectar con facilidad los conflictos de nombres entre funciones. Este parche agrega la macro al código fuente.

```
patch −Np1 −i
../findutils−4.1.patch
```

Hay un error en el fichero locate.c de Findutils que provoca una violación de segmento al encontrarse con rutas demasiado largas. El problema lo causa la función get_short() al calcular números negativos de forma incorrecta. Este parche soluciona dicho error.

```
patch −Np1 −i
../findutils−4.1−segfault.patch
```

Prepara Findutils para su compilación:

```
./configure −−prefix=/usr
```

Por defecto, la base de datos de updatedb se encuentra en /usr/var. Para hacer que la localización del fichero `locatedb` cumpla con el FHS, pásale la opción *localstatedir=/var/lib/misc* a los dos comandos **`make`** que vienen a continuación.

Continúa compilando el paquete:

```
make libexecdir=/usr/bin
```

Y termina instalando el paquete:

```
make libexecdir=/usr/bin
install
```

Instalación de Gawk–3.1.1

```
Estimación del tiempo de construcción:       0.39 SBU
Estimación del espacio necesario en disco:  15 MB
```

Contenido de Gawk

Última versión comprobada: 3.1.1.

Gawk es uns implementación de awk utilizada para manipular ficheros de texto.

Gawk instala lo siguiente:

Programas

awk, gawk, gawk–3.1.1, grcat, igawk, pgawk, pgawk–3.1.1, pwcat

Dependencias de instalación de Gawk

Última versión comprobada: 3.1.0.

(Todavía no se han comprobado las dependencias.)

Instalación de Gawk

Antes de instalar el paquete Gawk tienes que aplicarle un parche que corrige lo siguiente:

- Para Gawk, la ubicación por defecto de libexecdir es`$prefix/libexecdir/awk`. Esta ubicación no cumple con el FHS (el cual nunca menciona un directorio llamado libexecdir).
- El parche permite que le pasemos *--libexecdir* al guión configure (sin que gawk le clave /awk al final), para que podamos usar una ubicación más apropiada para el libexecdir de gawk (`/usr/bin` en el libro).
- El directorio de datos por defecto de gawk es `$prefix/share/awk`. Un directorio específico de un paquete tendría que ser nombrado usando el paquete y su versión (como gawk–3.1.1 en vez de awk) porque podría haber más de un intérprete de awk en el sistema (y más de una versión de gawk). El parche cambia este directorio por `$prefix/share/gawk-3.1.1` para que sea más correcto.
- Este parche también se asegura que dicho directorio (`$prefix/share/gawk-3.1.1`) y su contenido sean borrados al desinstalar gawk con make uninstall.

```
patch -Np1 -i
../gawk-3.1.1-3.patch
```

Prepara Gawk para su compilación:

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

Contunúa con la compilación del paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Ncurses–5.3

```
Estimación del tiempo de construcción:       1.88 SBU
Estimación del espacio necesario en disco:  22 MB
```

Contenido de Ncurses

Última versión comprobada: 5.2.

El paquete Ncurses proporciona librerías para el manejo de caractéres y terminales, incluidos paneles y menús.

Ncurses instala lo siguiente:

Programas

captoinfo (enlace a tic), clear, infocmp, infotocap (enlace a tic), reset (enlace a tset), tack, tic, toe, tput y tset.

Librerías

libcurses.[a,so] (enlace a libncurses.[a,so]), libform.[a,so], libform_g.a, libmenu.[a,so], libmenu_g.a, libncurses++.a, libncurses.[a,so], libncurses_g.a, libpanel.[a,so] y libpanel_g.a

Dependencias de instalación de Ncurses

Última versión comprobada: 5.2.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mkdir, mv, rm
Gcc: c++, cc1, cc1plus, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, fgrep, grep
Make: make

Gawk: gawk
Sed: sed
Sh−utils: basename, date, echo, expr, hostname, uname
Textutils: cat, sort, tr, wc

Instalación de Ncurses

Prepara Ncurses para su compilación:

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

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Otorga permisos de ejecución a las librerías ncurses:

```
chmod 755 /usr/lib/*.5.3
```

Mueve las librerías al directorio `/lib`, donde se espera encontrarlas:

```
mv /usr/lib/libncurses.so.5* /lib
```

Puesto que las librerías se han movido a `/lib`, algunos enlaces simbólicos apuntan ahora a ficheros que no existen. Regenera esos enlaces simbólicos:

```
ln -sf libncurses.a /usr/lib/libcurses.a &&
ln -sf ../../lib/libncurses.so.5 /usr/lib/libncurses.so &&
ln -sf ../../lib/libncurses.so.5 /usr/lib/libcurses.so
```

Instalación de Vim−6.1

```
Estimación del tiempo de construcción:       0.81 SBU
Estimación del espacio necesario en disco:  24 MB
```

Contenido de Vim

Última versión comprobada: 6.1.

El paquete Vim contiene un editor de texto configurable construido para obtener una eficiente edición del texto.

Vim instala lo siguiente:

Programas

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, tcltags, vi (enlace a vim), view (enlace a vim), vim, vim132, vim2html.pl, vimdiff (enlace a vim), vimm, vimspell.sh, vimtutor y xxd

Alternativas a Vim

- emacs, joe y nano

  http://www.escomposlinux.org/lfs-es/blfs-es-CVS/postlfs/editors.html (en castellano)

  http://beyond.linuxfromscratch.org/view/cvs/postlfs/editors.html (en inglés)

Dependencias de instalación de Vim

Última versión comprobada: 6.0.

Bash: sh
Binutils: as, ld, strip
Diffutils: cmp, diff
Fileutils: chmod, cp, ln, mkdir, mv, rm, touch
Find: find
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Net-tools: hostname
Sed: sed
Sh-utils: echo, expr, uname, whoami
Textutils: cat, tr, wc

Instalación de Vim

Este paquete necesita que le apliques un parche antes de poder instalarlo. Este parche corrige un problema de compilación con GCC-3.2:

```
patch -Np1 -i
../vim-6.1.patch
```

Prepara Vim para su compilación:

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

Continúa compilando el paquete:

```
make
CPPFLAGS=-DSYS_VIMRC_FILE=\\\"/etc/vimrc\\\"
```

El significado de la opción de make es:

- **CPPFLAGS=-DSYS_VIMRC_FILE=\\\"/etc/vimrc\\\":** Esta opción hace que vim busque el fichero `/etc/vimrc` que contiene las opciones globales de vim. Normalmente se asume que este fichero se encuentra en `/usr/share/vim`, pero creemos que `/etc` es un sitio más lógico para este tipo de ficheros.

Termina la instalación del paquete:

```
make install
```

Vim puede ejecutarse con el modo antiguo de *vi* mediante la creación de un enlace simbólico, que puede crearse con el siguiente comando:

```
ln -s vim /usr/bin/vi
```

Si piensas instalar el sistema X Window en tu sistema LFS, puede que quieras recompilar Vim después de instalar X. Vim tiene una bonita versión con interfaz gráfica que necesita X y alguna otra librería instalada. Para más información lee la documentación de Vim.

Configuración de Vim

Por defecto, vim se ejecuta en modo compatible con vi. Hay gente a la que le puede gustar esto, pero nosotros preferimos ejecutar vim en modo vim (de otra forma, no habríamos incluido vim en este libro, sino el vi original). Crea el fichero `/root/.vimrc` ejecutando lo siguiente:

```
cat > /root/.vimrc << "EOF"
# Inicio de /root/.vimrc

set nocompatible
set bs=2

# Fin de /root/.vimrc
EOF
```

Instalación de M4–1.4

```
Estimación del tiempo de construcción:       0.08 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido de M4

Última versión comprobada 1.4.

M4 es un procesador de macros. Copia la entrada a la salida expandiendo las macros en el proceso. Las macros pueden ser internas o definidas por el usuario y pueden tomar cualquier número de argumentos.

Aparte de hacer la expansión de macros, m4 tiene funciones internas para la inclusión de 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.

M4 instala lo siguiente:

Programas

m4

Dependencias de instalación de M4

Última versión comprobada: 1.4.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Make: make
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Sed: sed
Sh−utils: date, echo, hostname
Textutils: cat, tr

Instalación de M4

Prepara M4 para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Bison−1.875

```
Estimación del tiempo de construcción:       0.27 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Bison

Última versión comprobada: 1.35.

Bison es un generador de analizadores sintácticos, un sustituto de yacc. Bison genera un programa que analiza la estructura de un fichero de texto.

Bison instala lo siguiente:

Programas

bison y yacc

Dependencias de instalación de Bison

Última versión comprobada: 1.31.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, head, tr, uniq

Instalación de Bison

Prepara Bison para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Less–378

```
Estimación del tiempo de construcción:       0.13 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Less

Última versión comprobada: 374.

Less es un paginador de ficheros, o visor de texto. Muestra el contenido de un fichero, o de un flujo de datos, y tiene la habilidad de poder recorrerlo. Less tiene várias características no incluidas en el paginador **more**, como la habilidad de recorrer los ficheros hacia atrás.

Less instala lo siguiente:

Programas

less, lessecho y lesskey

Dependencias de instalación de Less

Última versión comprobada: 358.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, mv, rm, touch
Grep: egrep, grep
Gcc: cc1, collect2, cpp0, gcc
Make: make
Sed: sed
Sh–utils: expr, hostname, uname
Textutils: cat, tr

Instalación de Less

Prepara Less para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Groff–1.18.1

```
Estimación del tiempo de construcción:       1.08 SBU
Estimación del espacio necesario en disco:  18 MB
```

Contenido de Groff

Úlrima versión comprobada: 1.17.2.

El paquete Groff incluye varios programas de procesamiento de texto para formatear el texto. Grof traduce el texto estándar y los comandos especiales a salida formateada, como la que puedes ver en las páginas de manual.

Groff instala lo siguiente:

Programas

addftinfo, afmtodit, eqn, geqn (enlace a eqn), grn, grodvi, groff, grog, grolbp, grolj4, grops, grotty, gtbl (enlace a tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, post−grohtml, pre−grohtml, refer, soelim, tbl, tfmtodit, troff y zsoelim (enlace a soelim)

Dependencias de instalación de Groff

Última versión comprobada: 1.17.2.

Bash: sh
Binutils: ar, as, ld, ranlib
Bison: bison
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, touch
Gcc: cc1, cc1plus, collect2, cpp0, g++, gcc
Grep: egrep, grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, date, echo, expr, hostname, uname
Textutils: cat, tr

Instalación de Groff

Prepara Groff para su compilación:

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

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Algunos programas de documentación groff/man, como **xman**, no funcionarán correctamente sin los siguientes enlaces simbolicos.

```
ln -s soelim /usr/bin/zsoelim &&
ln -s eqn /usr/bin/geqn &&
ln -s tbl /usr/bin/gtbl
```

Instalación de Textutils–2.1

```
Estimación del tiempo de construcción:       0.83 SBU
Estimación del espacio necesario en disco:  17 MB
```

Contenido deTextutils

Última versión comprobada: 2.0.

El paquete Textutils contiene varios programas para procesar y manipular ficheros de texto.

Textutils instala lo siguiente:

Programas

cat, cksum, comm, csplit, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq y wc

Dependencias de instalación de Textutils

Última versión comprobada: 2.0.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp

Fileutils: chmod, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Textutils

Prepara Textutils para su compilación:

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

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Finalmente, completa la instalación de este paquete moviendo algunos de sus programas a un directorio más apropiado:

```
mv /usr/bin/{cat,head} /bin
```

Instalación de Sed−4.0.5

```
Estimación del tiempo de construcción:       0.09 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Sed

Última versión comprobada: 3.02.

sed es un editor de flujo. Un editor de flujo se utiliza para realizar transformaciones básicas de texto sobre un flujo de entrada (un fichero o la entrada procedente de una tubería).

Sed instala lo siguiente:

Programas

sed

Dependencias de instalación de Sed

Última versión comprobada: 3.02.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: echo, expr, hostname, sleep
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Sed

Prepara Sed para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Flex−2.5.4a

```
Estimación del tiempo de construcción:       0.05 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido de Flex

Última versión comprobada: 2.5.4a.

El pquete Flex se utiliza para generar programas que reconocen patrones de texto.

Flex instala lo siguiente:

Programas

flex, flex++ (enlace a flex) y lex

Librerías

libfl.a

Dependencias de instalación de Flex

Última versión comprobada: 2.5.4a.

Bash: sh
Binutils: ar, as, ld, ranlib
Bison: bison
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: echo, hostname
Textutils: cat, tr

Instalación de Flex

Prepara Flex para su compilación:

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

Continúa compilando el paquete:

```
make
```

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 -s libfl.a /usr/lib/libl.a
```

Algunos programas no conocen flex e intentan encontrar el programa lex (flex es una alternativa [mejor] para lex). Para complacer a estos programas, crea un guión de nombre lex que llame a flex en modo de emulación:

```
cat > /usr/bin/lex << "EOF"
#!/bin/sh
# Inicio de /usr/bin/lex

exec /usr/bin/flex -l "$@"

# Fin de /usr/bin/lex
EOF
chmod 755 /usr/bin/lex
```

Instalación de Binutils–2.13.2

```
Estimación del tiempo de construcción:       2.48 SBU
Estimación del espacio necesario en disco:  94 MB
```

Contenido de Binutils

Última versión comprobada: 2.12.1.

Binutils es una colección de herramientas para el desarrollo de software que contiene un enlazador, un ensamblador y otras utilidades para trabajar con ficheros de objetos y archivos.

Binutils instala lo siguiente:

Programas

addr2line, ar, as, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings y strip

Librerías

libbfd.[a,so] y libopcodes.[a,so]

Dependencias de instalación de Binutils

Última versión comprobada: 2.11.2.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh

Binutils: ar, as, ld, nm, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, rmdir, touch
Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, true, uname
Texinfo: install−info, makeinfo
Textutils: cat, sort, tr, uniq

Instalación de Binutils

Se sabe que este programa se comporta mal si cambias sus parámetros de optimización (incluyendo las opciones −march y −mcpu). Por tanto, si tienes definida cualquier variable de entorno que pueda sobreescribir las optimizaciones por defecto, como CFLAGS y CXXFLAGS, te recomendamos que las desactives o modifiques antes de construir Binutils.

La documentación sobre la instalación de Binutils recomienda construir Binutils fuera del directorio de las fuentes:

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

A continuación, prepara Binutils para su compilación:

```
../binutils-2.13.2/configure --prefix=/usr --enable-shared
```

Continúa compilando el paquete:

```
make tooldir=/usr
```

Normalmente, el directorio **tooldir** (donde se instalarán los ejecutables de Binutils) se establece como $(exec_prefix)/$(target_alias), lo que se convierte en, por ejemplo, /usr/i686−pc−linux−gnu. Como sólo construimos programas para nuestro propio sistema, no necesitamos en /usr este directorio específico de cada objetivo. Esa configuración se utilizaría si el sistema fuese usado para compilación cruzada (por ejemplo, compilando un paquete en una máquina Intel, pero generando código que se ejecutará en máquinas Apple PowerPC).

Instala el paquete:

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

Instala las páginas info:

```
make tooldir=/usr install-info
```

Algunos paquetes necesitan la cabecera *libiberty* para poder construirse. Para satisfacer a estos programas, instala el fichero:

```
cp ../binutils-2.13.2/include/libiberty.h /usr/include
```

Instalación de Fileutils–4.1

```
Estimación del tiempo de construcción:       0.68 SBU
Estimación del espacio necesario en disco:  17 MB
```

Contenido de Fileutils

Última versión comprobada: 4.1.

Fileutils es un paquete que contiene los programas básicos para la manipulación de ficheros. Incluye programas para listar y crear directorios, actualizar las marcas de fechas, cambiar los permisos y más.

Fileutils instala lo siguiente:

Programas

chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, shred, sync, touch y vdir

Dependencias de instalación de Fileutils

Última versión comprobada: 4.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Perl: perl
Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install–info
Textutils: cat, tr

Instalación de Fileutils

Prepara Fileutils para su compilación:

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

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Algunos paquetes tienen fijada la ruta al programa **install** como */usr/bin/install*. Crea un enlace simbólico para cumplir con esto:

```
ln -s ../../bin/install /usr/bin
```

Instalación de Sh-utils-2.0

```
Estimación del tiempo de construcción:       0.42 SBU
Estimación del espacio necesario en disco:  12 MB
```

Contenido de Sh-utils

Última versión comprobada: 2.0.

El paquete Sh-utils contiene un número de utilidades para realizar manipulaciones básicas en el intérprete de comandos.

Sh-utils instala lo siguiente:

Programas

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami y yes

Dependencias de instalación de Sh-utils

Última versión comprobada: 2.0.

Autoconf: autoconf, autoheader
Automake: aclocal, automake

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, chown, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Sh−utils

Este paquete necesita que le apliques el parche de hostname antes de poder instalarlo. Este parche suprime la construcción del programa hostname, que será instalado posteriormente con el paquete net−tools. El programa hostname del paquete net−tools es una versión mucho mejor (y, en algunos casos, necesaria, pues soporta opciones que necesitan algunos programas como XFree86). Aplica el parche:

```
patch -Np1 -i ../sh-utils-2.0-hostname.patch
```

Prepara Shellutils para su compilación:

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

Continúa compilado el paquete:

```
make
```

Instala el paquete:

```
make install
```

Finalmente, mueve algunos de los programas a una localización más adecuada:

```
mv /usr/bin/{basename,date,echo,false,pwd} /bin &&
mv /usr/bin/{sleep,stty,su,test,true,uname} /bin &&
mv /usr/bin/chroot /usr/sbin
```

Notas sobre la conformidad con el estándar FHS

Hay un comando que se instala con este paquete que se llama test. Se usa a menudo en guiones del intérprete de comandos (shell scripts) para evaluar condiciones, pero habitualmente se encuentra como **[condición]**. Estos corchetes son comandos internos del intérprete bash. Sin embargo, el estándar FHS determina que debe haber un programa llamado `[`. Créalo ejecutando:

```
ln -s test /bin/[
```

Instalación de Gettext–0.11.5

```
Estimación del tiempo de construcción:       0.99 SBU
Estimación del espacio necesario en disco:  39 MB
```

Contenido de Gettext

Última versión comprobada: 0.11.2.

El paquete Gettext se utiliza para la internacionalización y localización. Los programas pueden compilarse con Soporte de Lenguaje Nativo (NLS), lo que les permite mostrar mensajes en el idioma nativo del usuario.

Gettext instala lo siguiente:

Programas

config.charset, config.rpath, gettext, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, project–id, team–address, trigger, urlget, user–email y xgettext

Librerías

libgettextlib[a,so], libgettextsrc[a,so]

Dependencias de instalación de Gettext

Última versión comprobada: 0.10.40.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip
Bison: bison
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm, rmdir
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
M4: m4

Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, sort, tr, uniq

Instalación de Gettext

Prepara Gettext para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Net−tools−1.60

```
Estimación del tiempo de construcción:       0.16 SBU
Estimación del espacio necesario en disco:  5 MB
```

Contenido de Net−tools

Última versión comprobada: 1.60.

El paquete Net−tools contiene una colección de programas que forman la base del trabajo en red en Linux.

Net−tools instala lo siguiente:

Programas

arp, dnsdomainname (enlace a hostname), domainname (enlace a hostname), hostname, ifconfig, nameif, netstat, nisdomainname (enlace a hostname), plipconfig, rarp, route, slattach y ypdomainname (enlace a hostname)

Dependencias de instalación de Net−tools

Última versión comprobada: 1.60.

Bash: bash, sh
Binutils: ar, as, ld
Fileutils: install, ln, ls, mv, rm
Gcc: cc, cc1, collect2, cpp0
Make: make
Sh−utils: echo

Instalación de Net−tools

Si no sabes qué contestar a todas las preguntas que se hacen durante la etapa **make**, entonces basta con aceptar los valores por defecto, ya que será lo correcto en la mayoría de los casos. Lo que se pregunta aquí es una serie de cuestiones relativas al tipo de protocolos de red que tienes activados en tu núcleo.

Las respuestas por defecto activarán las herramientas de este paquete para trabajar con la mayoría de los protocolos más comunes, como TCP, PPP y algunos otros. En realidad, todavía necesitarás activar esos protocolos en el núcleo. Lo que estás haciendo aquí es, simplemente, ordenar a los programas que sean capaces de usar esos protocolos, pero corre por cuenta del núcleo dejarlos disponibles para el sistema.

Compila el paquete:

```
make
```

Si quieres aceptar todas las respuestas que se ofrecen por defecto, puedes saltarte las preguntas planteadas por *make* ejecutando **yes "" | make**.>

Y termina instalando el paquete:

```
make update
```

La opción **update** para **make** hace lo mismo que la opción **install**, con una excepción: no hace copias de seguridad de los ficheros que está reemplazando.

Si decides volver a instalar este paquete en algún momento, **make update** no hará copias de seguridad de todos los ficheros de la instalación anterior de net−tools.

Instalación de Perl−5.8.0

```
Estimación del tiempo de construcción:       3.81 SBU
Estimación del espacio necesario en disco:  52 MB
```

Contenido de Perl

Última versión comprobada: 5.6.1.

El paquete Perl contiene perl, el Lenguaje Práctico de Extracción e Informe. Perl combina alguna de las mejores características de C, sed, awk y sh dentro de un poderoso lenguaje.

Perl instala lo siguiente:

Programas

a2p, c2ph, dprofpp, find2perl, h2ph, h2xs, perl, perl5.6.1, perlbug, perlcc, perldoc, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, pstruct, s2p y splain

Librerías

attrs.so, B.so, ByteLoader.so, DProf.so, Dumper.so, DynaLoader.a, Fcntl.so, Glob.so, Hostname.so, IO.so, libperl.a, Opcode.so, Peek.so, POSIX.so, re.so, SDBM_File.so, Socket.so, Syslog.so y SysV.so

Dependencias de instalación de Perl

Última versión comprobada: 5.6.1.

Bash: sh
Binutils: ar, as, ld, nm
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, date, echo, expr, hostname, pwd, uname, whoami
Textutils: cat, comm, sort, split, tr, uniq, wc

Instalación de Perl

Prepara Perl para su compilación:

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

Si quieres más control sobre la forma en que perl se autoconfigura para construirse, puedes ejecutar en su lugar el guión interactivo **`Configure`** y modificar el modo en que perl se construye. Si piensas que puedes vivir con los (razonables) valores por defecto que perl autodetecta, entonces usa los comandos listados anteriormente.

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Texinfo–4.3

```
Estimación del tiempo de construcción:       0.43 SBU
Estimación del espacio necesario en disco:  12 MB
```

Contenido de Texinfo

Última versión comprobada: 4.2.

El paquete Texinfo contiene programas usados para leer, escribir y convertir documentos Info, que suministran documentación del sistema.

Texinfo instala lo siguiente:

Programas

info, infokey, install–info, makeinfo, texi2dvi y texindex

Dependencias de instalación de Texinfo

Última versión comprobada: 4.0.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Sed: sed
Sh–utils: basename, echo, expr, hostname, sleep
Texinfo: makeinfo
Textutils: cat, tr

Intalación de Texinfo

Prepara Texinfo para su compilación:

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

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Instala los componentes de texinfo que pertenecen a una instalación de TeX:

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

Instalación de Autoconf–2.57

```
Estimación del tiempo de construcción:       0.05 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Autoconf

Última versión comprobada: 2.53.

Autoconf produce guiones del interprete de comandos que configuran automáticamente el código fuente.

Autoconf instala lo siguiente:

Programas

autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames

Dependencias de instalación de Autoconf

Última versión comprobada: 2.52.

Bash: sh
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Grep: fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh–utils: echo, expr, hostname, sleep, uname
Texinfo: install–info
Textutils: cat, tr

Instalación de Autoconf

Prepara Autoconf para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Automake–1.7.2

```
Estimación del tiempo de construcción:       0.03 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Automake

Última versión comprobada: 1.6.2.

Automake genera ficheros Makefile.in, pensados para usar con Autoconf.

Automake instala lo siguiente:

Programas

acinstall, aclocal, aclocal–1.6, automake, automake–1.6, compile, config.guess, config.sub, depcomp, elisp–comp, install–sh, mdate–sh, missing, mkinstalldirs, py–compile, ylwrap

Dependencias de instalación de Automake

Última versión comprobada: 1.5.

Bash: sh
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm, rmdir
Grep: fgrep, grep
Make: make
Perl: perl
Sed: sed
Sh–utils: echo, expr, hostname, sleep
Texinfo: install–info
Textutils: cat, tr

Instalación de Automake

Prepara Automake para su compilación:

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

Continúa compilando e instalando el paquete:

```
make install
```

Crea los enlaces simbólicos necesarios:

```
ln -s automake-1.7 /usr/share/automake
```

Instalación de Bash–2.05a

```
Estimación del tiempo de construcción:       0.82 SBU
Estimación del espacio necesario en disco:  14 MB
```

Contenido de Bash

Última versión comprobada: 2.05a.

bash es la "Bourne−Again SHell", que es un completo intérprete de comandos usado ampliamente en sistemas Unix. El programa bash lee de la entrada estándar (el teclado). Un usuario escribe algo y el programa evalúa lo que ha escrito y hace algo con ello, como lanzar un programa.

Bash instala lo siguiente:

Programas

bash, sh (enlace a bash) y bashbug

Dependencias de instalación de Bash

Última versión comprobada: 2.05a.

Bash: bash, sh
Binutils: ar, as, ld, ranlib, size
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname

Texinfo: install−info
Textutils: cat, tr, uniq

Instalación de Bash

Prepara Bash para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Recarga el Bash recién compilado:

```
exec /bin/bash --login
```

Instalación de File−3.39

```
Estimación del tiempo de construcción:       0.21 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de File

Última versión comprobada: 3.39.

File es una utilidad usada para determinar el tipo de los ficheros.

File instala lo siguiente:

Programas

file

Dependencias de instalación de File

Última versión comprobada: 3.37.

Autoconf: autoconf, autoheader
Automake: aclocal, automake

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: echo, expr, hostname, sleep
Texinfo: makeinfo
Textutils: cat, tr

Instalación de File

Prepara File para su compilación:

```
./configure --prefix=/usr --datadir=/usr/share/misc
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Libtool−1.4.3

```
Estimación del tiempo de construcción:       0.15 SBU
Estimación del espacio necesario en disco:  7 MB
```

Contenido de Libtool

Última versión comprobada: 1.4.2.

GNU libtool es un guión para soporte genérico de librerías. Libtool oculta la complejidad del uso de librerías compartidas tras una interfaz consistente y portable.

Libtool instala lo siguiente:

Programas

libtool y libtoolize

Librerías

libltdl.a, libltdl.so (enlace a libltdl.so.3.1.0), libltdl.so.3 (enlace a libltdl.so.3.1.0) y libltdl.so.3.1.0

Dependencias de instalación de Libtool

Última versión comprobada: 1.4.2.

Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gcc: cc, cc1, collect2, cpp0
Glibc: ldconfig
Grep: egrep, fgrep, grep
Make: make
Sed: sed
Sh−utils: echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, sort, tr, uniq

Instalación de Libtool

Prepara Libtool para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Bin86−0.16.3

```
Estimación del tiempo de construcción:       0.07 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Bin86

Última versión comprobada: 0.16.3

Bin86 es un simple ensamblador y enlazador para código de máquinas 8086 – 80386.

Bin86 instala lo siguiente:

Programas

as86, as86_encap, ld86, nm86 (enlace a objdump86), objdump86 y size86 (enlace a objdump86)

Dependencias de instalación de Bin86

Última versión comprobada: 0.16.0.

Bash: sh
Binutils: as, ld, strip
Fileutils: chmod, install, ln, mv
Gcc: cc, cc1, collect2, cpp0
Make: make
Sed: sed

Instalación de Bin86

Este paquete sólo es necesario si decides utilizar Lilo en tu sistema LFS. Si vas a usar alguna otra utilidad, como Grub, no necesitas bin86. Revisa la documentación de tu gestor de arranque favorito para ver si necesita el paquete bin86 (normalmente, sólo son necesarios ld86 y/o as86).

Recuerda que no sólo los gestores de arranque requieren el paquete bin86. Siempre existe la posibilidad de que algún otro paquete necesite estos programas, así que tenlo en cuenta si decides saltarte este paquete.

Compila el paquete:

```
make
```

Y termina instalando el paquete:

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

Instalación de Bzip2–1.0.2

```
Estimación del tiempo de construcción:       0.09 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido de Bzip2

Última versión comprobada: 1.0.2

Bzip2 es un compresor de ficheros por ordenación de bloques que, generalmente, consige una mejor compresión que el tradicional **gzip**.

Bzip2 instala lo siguiente:

Programas

bunzip2 (enlace a bzip2), bzcat (enlace a bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless y bzmore

Librerías

libbz2.a, libbz2.so (enlace a libbz2.so.1.0), libbz2.so.1.0 (enlace a libbz2.so.1.0.2) y libbz2.so.1.0.2

Dependencias de instalación de Bzip2

Última versión comprobada: 1.0.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Fileutils: cp, ln, rm
Gcc: cc1, collect2, cpp0, gcc
Make: make

Instalación de Bzip2

Compila el paquete:

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

La opción **-f** 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.

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Copia el binario de Bzip2 al directorio `/bin`, crea algunos enlaces simbólicos necesarios y haz limpieza.

```
cp bzip2-shared /bin/bzip2 &&
cp -a libbz2.so* /lib &&
ln -s ../../lib/libbz2.so.1.0 /usr/lib/libbz2.so &&
rm /usr/bin/{bunzip2,bzcat,bzip2} &&
mv /usr/bin/{bzip2recover,bzless,bzmore} /bin &&
ln -s bzip2 /bin/bunzip2 &&
ln -s bzip2 /bin/bzcat
```

Instalación de Ed-0.2

```
Estimación del tiempo de construcción:       0.06 SBU
Estimación del espacio requierido en disco:  3 MB
```

Contenido de Ed

Última versión comprobada 0.2.

GNU ed es un editor de líneas de 8 bits limpio y que cumple con POSIX.

Ed instala lo siguiente:

Programas

ed y red (enlace a ed)

Dependencias de instalación de Ed

Última versión comprobada: 0.2.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh-utils: hostname
Textutils: cat, tr

Instalación de Ed

> **Nota:** Ed no es algo que utilice mucha gente. Se instala aquí porque puede que lo use el programa patch si te encuentras con algún parche basado en ed. Esto no suele ocurrir porque ahora se prefieren los parches basados en diff.

Ed usa mktemp para crear ficheros temporales en `/tmp`, pero esta función tiene una vulnerabilidad de seguridad (ver la sección Temporary Files en http://en.tldp.org/HOWTO/Secure−Programs−HOWTO/avoid−race.html). Este parche hace que Ed use mkstemp, que es la forma recomendada para crear ficheros temporales.

Aplica el parche:

```
patch -Np1 -i
../ed-0.2.patch
```

Prepara Ed para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Necesitamos mover los binarios de Ed al directorio `/bin`, pues deben poder usarse en caso de que la partición `/usr` no esté disponible.

```
mv /usr/bin/{ed,red} /bin
```

Instalación de Kbd−1.08

```
Estimación del tiempo de construcción:       0.12 SBU
Estimación del espacio necesario en disco:  8 MB
```

Contenido de Kbd

Última versión comprobada: 1.06.

Kbd contiene ficheros de mapas de teclado y utilidades para el teclado.

Kbd instala lo siguiente:

Programas

chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, getunimap, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (enlace a psfxtable), psfgettable (enlace a psfxtable), psfstriptable (enlace a psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setlogcons, setmetamode, setvesablank,

showfont, showkey, unicode_start, y unicode_stop

Dependencias de instalación de Kbd

Última versión comprobada: 1.06.

Bash: sh
Binutils: as, ld, strip
Bison: bison
Diffutils: cmp
Fileutils: cp, install, ln, mv, rm
Flex: flex
Gettext: msgfmt, xgettext
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Gzip: gunzip, gzip
Make: make
Patch: patch
Sed: sed
Sh−utils: uname

Instalación de Kbd

Kbd no instala algunas de sus utilidades (setlogcons, setvesablank y getunimap) por defecto. El parche para kbd activa la compilación de estas utilidades. Aplica el parche:

```
patch -Np1 -i ../kbd-1.08.patch
```

Prepara Kbd para su compilación:

```
./configure
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Diffutils−2.8.1

```
Estimación del tiempo de construcción:       0.31 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Diffutils

Última versión comprobada: 2.8.1.

Los programas de este paquete te muestran las diferencias entre dos ficheros o directorios. Es muy común usarlos para crear parches de software.

Diffutils instala lo siguientes:

Programas

cmp, diff, diff3 y sdiff

Dependencias de instalación de Diffutils

Última versión comprobada: 2.7.

Bash: sh
Binutils: ld, as
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: date, hostname
Textutils: cat, tr

Instalación de Diffutils

Prepara Diffutils para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de E2fsprogs−1.32

```
Estimación del tiempo de construcción:       0.80 SBU
Estimación del espacio necesario en disco:  13 MB
```

Contenido de E2fsprogs

Última versión comprobada: 1.27.

E2fsprogs proporciona las utilidades para los sistemas de ficheros ext2. También soporta los sistemas de ficheros ext3 con registro de transaciones.

E2fsprogs instala lo siguiente:

Programas

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs y uuidgen

Librerías

libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], libuuid.[a,so]

Dependencias de instalación de E2fsprogs

Última versión comprobada: 1.25.

Bash: sh
Binutils: ar, as, ld, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mkdir, mv, rm, sync
Gcc: cc, cc1, collect2, cpp0
Glibc: ldconfig
Grep: egrep, grep
Gzip: gzip
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, echo, expr, hostname, uname
Texinfo: makeinfo
Textutils: cat, tr

Instalación de E2fsprogs

Se recomienda construir E2fsprogs fuera del árbol de las fuentes:

```
mkdir ../e2fsprogs-build &&
cd ../e2fsprogs-build
```

Prepara E2fsprogs para su compilación:

```
../e2fsprogs-1.32/configure
--prefix=/usr --with-root-prefix="" \
    --enable-elf-shlibs
```

El significado de las opciones de configure es:

- **`--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 le pasaramos esta opción al configure de E2fsprogs, los programas se colocarían en el directorio `/usr`, que no es lo que queremos.
- **`--enable-elf-shlibs:`** Esto crea las librerías compartidas utilizadas por algunos de los programas de este paquete.

Continúa compilando el paquete:

```
make
```

Comienza la instalación del paquete:

```
make install
```

Instala las librerías compartidas:

```
make install-libs
```

Actualiza el fichero `/usr/share/info/dir` para incluir las páginas info de E2fsprogs en el índice:

```
install-info /usr/share/info/libext2fs.info
/usr/share/info/dir
```

Instalación de Grep–2.5

```
Estimación del tiempo de construcción:       0.22 SBU
Estimación del espacio necesario en disco:  5 MB
```

Contenido de Grep

Última versión comprobada: 2.5.

Grep es un programa usado para imprimir las líneas de un fichero que cumplan un patrón especificado.

Grep instala lo siguiente:

Programas

egrep (enlace a grep), fgrep (enlace a grep) y grep

Dependencias de instalación de Grep

Última versión comprobada: 2.4.2.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Grep

Prepara Grep para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Gzip−1.2.4a

```
Estimación del tiempo de construcción:       0.03 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Gzip

Última versión comprobada: 1.2.4a.

El paquete Gzip contiene programas para comprimir y descomprimir ficheros usando el codificador Lempel−Ziv (LZ77).

Gzip instala lo Siguiente:

Programas

gunzip (enlace a gzip), gzexe, gzip, uncompress (enlace a gunzip), zcat (enlace a gzip), zcmp, zdiff, zforce, zgrep, zmore y znew

Dependencias de instalación de Gzip

Última versión comprobada: 1.2.4a.

Bash: sh
Binutils: as, ld, nm
Fileutils: chmod, cp, install, ln, mv, rm
Gcc: cc1, collect2, cpp, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: hostname
Textutils: cat, tr

Instalación de Gzip

El siguiente parche corrige un desbordamiento de la memoria intermedia (buffer overflow) que ocurre cuando un nombre de fichero supera los 1020 caracteres. Esto se soluciona verificando que la memoria intermedia (buffer) es lo suficientemente grande para dicho nombre de fichero. El programa termina con el mensaje "Filename too long" ("Nombre de fichero demasiado largo") si la memoria intermedia no es lo suficientemente grande.

```
patch -Np1 -i
../gzip-1.2.4b.patch
```

Prepara Gzip para su compilación:

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

Cambia el directorio de instalación por defecto de Gzip, pues será instalado en el directorio `/bin`:

```
cp gzexe.in{,.backup} &&
sed 's%"BINDIR"%/bin%' gzexe.in.backup > gzexe.in
```

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Mueve los binarios de Gzip al directorio `/bin`:

```
mv /usr/bin/gzip /bin &&
rm /usr/bin/{gunzip,zcat} &&
ln -s gzip /bin/gunzip &&
ln -s gzip /bin/zcat &&
ln -s gunzip /bin/uncompress
```

Instalación de Man-1.5k

```
Estimación del tiempo de construcción:       0.05 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Man

Última versión comprobada: 1.5k.

Man es un paginador de manuales.

Man instala lo siguiente:

Programas

apropos, makewhatis, man, man2dvi, man2html y whatis

Dependencias de instalación de Man

Última versión comprobada: 1.5i.

Bash: sh
Binutils: as, ld
Fileutils: chmod, cp, install, mkdir, rm
Gcc: c11, collect2, cpp0, gcc
Grep: grep

Make: make
Gawk: awk
Sed: sed
Sh−utils: echo
Textutils: cat

Instalación de Man

Hay tres parches para Man. El primer parche comenta una de las líneas del fichero `man.conf` (MANPATH /usr/man) para evitar resultados redundantes cuando utilicemos programas como **`whatis`**:

```
patch -Np1 -i ../man-1.5k-manpath.patch
```

El segundo parche añade la opción *−R* a la variable *PAGER* para que las secuencias de escape se manejen correctamente:

```
patch -Np1 -i ../man-1.5k-pager.patch
```

El último parche evita problemas cuando las páginas de manual que no están formateadas para más de 80 columnas se usan en conjunto con las versiones recientes de **`groff`**:

```
patch -Np1 -i ../man-1.5k-80cols.patch
```

Las rutas a algunos programas se escriben dentro de los ficheros de man. Desafortunadamente, el guión configure asigna la última localización en el PATH en la que se encuentra un programa, en lugar de la primera. Añadiendo */usr/bin:/bin* a la variable PATH para el comando **`./configure`** nos aseguramos de que man no utilice los programas del directorio `/static`.

Prepara Man para su compilación:

```
PATH=$PATH:/usr/bin:/bin \
    ./configure -default -confdir=/etc
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

> **Nota:** Si deseas desactivar las secuencias de escape SGR, debes editar el fichero man.conf y añadir el argumento **`-c`** a nroff.

Puede que quieras mirar la receta sobre man en http://www.escomposlinux.org/lfs−es/recetas/man.html (el original se encuentra en http://hints.linuxfromscratch.org/hints/man.txt) que se ocupa de las cuestiones de formateado y compresión de las páginas de manual.

Instalación de Lilo–22.2

```
Estimación del tiempo de construcción:       0.08 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido de Lilo

Última versión comprobada: 22.2.

Lilo es el LInux LOader, cargador de Linux.

Lilo instala lo siguiente:

Programas

lilo, mkrescue y keytab–lilo.pl

Dependencias de instalación de Lilo

Última versión comprobada: 22.1.

Bash: sh
Bin86: as86, ld86
Binutils: as, ld, strip
Fileutils: cp, dd, ln
Gcc: cc, cc1, collect2, cpp0
Make: make
Sed: sed
Textutils: cat

Instalación de Lilo

Hemos elegido Lilo como gestor de arranque porque nos sentimos cómodos con él, pero puede que desees elegir otro. Fabio Fracassi ha escrito una receta sobre GRUB, que está disponible en http://www.escomposlinux.org/lfs–es/recetas/grub–howto.html (la versión original se encuentra en http://hints.linuxfromscratch.org/hints/grub–howto.txt).

Compila Lilo:

```
make
```

Y termina instalando el paquete:

```
make install
```

Al final de la instalación, aparecerá un mensaje afirmando que se debe ejecutar `/sbin/lilo`. Esto es inutil porque el fichero `/etc/lilo.conf` no está presente todavía. Completaremos la instalación de lilo en el Capítulo 8.

El símbolo estándar de LILO, o el menú, pueden sustituirse por el logo de LFS, o cualquier logo que desees. Martin Imobersteg ha escrito una receta sobre esto, que se encuentra en http://www.escomposlinux.org/lfs−es/recetas/bootlogo.html (la versión original se encuentra en http://hints.linuxfromscratch.org/hints/bootlogo.txt).

Instalación de Make−3.80

```
Estimación del tiempo de construcción:       0.22 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Make

Última versión comprobada: 3.79.1.

Make determina, automáticamente, qué piezas de un programa largo es necesario recompilar y ejecuta los comandos para recompilarlas.

Make instala lo siguiente:

Programas

make

Dependencias de instalación de Make

Última versión comprobada: 3.79.1.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chgrp, chmod, install, ls, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Make

Prepara Make para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Por defecto, `/usr/bin/make` se instala con kmem como grupo efectivo de ejecución (setgid). Esto es necesario en algunos sistemas para que pueda comprobar la carga media del sistema utilizando `/dev/kmem`. Sin embargo, en los sistemas Linux el permiso para el grupo kmem no es necesario, así que podemos quitar el bit SGID del programa `make`. Esto también soluciona algunos problemas que hacen que `make` ignore ciertas variables, como *LD_LIBRARY_PATH*.

```
chgrp root /usr/bin/make &&
chmod 755 /usr/bin/make
```

Instalación de Modutils–2.4.22

```
Estimación del tiempo de construcción:       0.13 SBU
Estimación del espacio necesario en disco:  3 MB
```

Contenido de Modutils

Última versión comprobada: 2.4.16.

El paque Modutils contiene programas que puedes utilizar para trabajar con los módulos del núcleo.

Modutils instala lo siguiente:

Programas

depmod, genksyms, insmod, insmod_ksymoops_clean, kallsyms (enlace a insmod), kernelversion, ksyms (enlace a insmod), lsmod (enlace a insmod), modinfo, modprobe (enlace a insmod) y rmmod (enlace a insmod)

Dependencias de instalación de Modutils

Última versión comprobada: 2.4.12.

Bash: sh
Binutils: ar, as, ld, ranlib, strip
Bison: bison
Diffutils: cmp
Fileutils: chmod, install, ln, mkdir, mv, rm
Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: basename, expr, hostname, uname
Textutils: cat, tr

Instalación de Modutils

Prepara Modutils para su compilación:

```
./configure
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Netkit−base−0.17

```
Estimación del tiempo de construcción:       0.03 SBU
Estimación del espacio necesario en disco:  1 MB
```

Contenido de Netkit−base

Última versión comprobada: 0.17.

El paquete Netkit−base contiene dos herramientas de red: ping, que determina el tiempo de respuesta de los paquetes ICMP, y el demonio inetd, que escucha las peticiones a los puertos y asigna estas peticiones a los programas adecuados.

Netkit−base instala lo siguiente:

Programas

inetd y ping

Dependencias de instalación de Netkit−base

Última versión comprobada: 0.17.

Bash: sh
Binutils: as, ld, strip
Fileutils: cp, install, rm
Make: make
Gcc: cc1, collect2, cpp0, gcc
Sed: sed
Sh−utils: date
Textutils: cat

Instalación de Netkit−base

Prepara Netkit−base para su compilación:

```
./configure
```

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Finalmente, dos ficheros de configuración esenciales deben instalarse en el directorio `/etc`. Hay otros ficheros en el directorio `etc.sample` que podrían interesarte.

Completa la instalación de este paquete:

```
cp etc.sample/{services,protocols} /etc
```

Instalación de Patch−2.5.4

```
Estimación del tiempo de construcción:       0.10
Estimación del espacio necesario en disco:  2 MB
```

Contenido dePatch

Última versión comprobada: 2.5.4.

El programa patch modifica un fichero basandose en un parche. Normalmente un parche es una lista, creada por el programa diff, que contiene instrucciones sobre cómo debe modificarse el fichero original.

Patch instala lo siguiente:

Programas

patch

Dependencias de instalación de Patch

Última versión comprobada: 2.5.4.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: echo, expr, hostname, uname
Textutils: cat, tr

Instalación de Patch

Prepara Patch para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Procinfo−18

```
Estimación del tiempo de construcción:       0.02 SBU
Estimación del espacio necesario en disco:  168 KB
```

Contenido de Procinfo

Última versión comprobada: 18.

El programa procinfo obtiene datos del sistema, como el uso de la memoria y los números de las interrupciones (IRQ), a partir del directorio `/proc`, y formatea estos datos de una forma atractiva.

Procinfo instala lo siguiente:

Programas

lsdev, procinfo y socklist

Dependencias de instalación de Procinfo

Última versión comprobada: 18.

Binutils: as, ld
Fileutils: install, mkdir
Gcc: cc1, collect2, cpp0, gcc
Make: make

Instalación de Procinfo

Compila Procinfo:

```
make LDLIBS=-lncurses
```

La opción *–lncurses* sobreescribe la opción por defecto, *–ltermcap*. Esto se hace debido a que se ha declarado como obsoleta a libtermcap a favor de libncurses.

Y termina instalando el paquete:

```
make install
```

Instalación de Procps–3.1.5

```
Estimación del tiempo de construcción:       0.14 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Procps

Última versión comprobada: 2.0.7.

El paquete Procps proporciona programas para supervisar y parar procesos del sistema. Procps obtiene la información sobre los procesos a través del directorio `/proc`.

Procps instala lo siguiente:

Programas

free, kill, oldps, pgrep, pkill, ps, skill, snice, sysctl, tload, top, vmstat, w y watch

Librerías

libproc.so

Dependencias de instalación de Procps

Última versión comprobada: 2.0.7.

Bash: sh
Binutils: as, ld, strip
Fileutils: install, ln, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, pwd
Textutils: sort, tr

Instalación de Procps

Este paquete necesita que le apliques un parche antes de poder instalarlo. Este parche corrige un problema con locale que hace que **w** falle bajo ciertos ajustes de las locales. Aplica el parche:

```
patch -Np1 -i ../procps-3.1.5.patch
```

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make XSCPT="" install
```

La opción **XSCPT** asignará un valor vacío a la variable de Makefile XSCPT, lo cual deshabilita la instalación de XConsole. De otro modo, **make install** intentaría copiar el fichero XConsole en /usr/X11R6/lib/X11/app−defaults. Ese directorio no existe, pues el entorno X no está instalado.

Instalación de Psmisc−21.2

```
Estimación del tiempo de construcción:       0.11 SBU
Estimación del espacio necesario en disco:  2 MB
```

Contenido de Psmisc

Última versión comprobada: 21.

El paquete Psmisc contiene tres programas que ayudan en el manejo del directorio `/proc`.

Psmisc instala lo siguiente:

Programas

fuser, killall y pstree

Dependencias de instalación de Psmisc

Última versión comprobada: 20.2

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Bison: bison
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: makeinfo
Textutils: cat, tr

Instalación dePsmisc

Prepara Psmisc para su compilación:

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

La opción *--exec-prefix=/* provocará que los programas se instalen en `/bin` en vez de en `/usr/bin`. Los programas de este paquete se usan a menudo en los guiones de inicio, así que deben estar en el directorio `/bin` para que puedan utilizarse cuando la partición `/usr` no se haya montado aún.

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

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.

Es hora de decidir si vas a utilizar el paquete Sysvinit, que contiene un programa pidof, o no. Si no vas a usar Sysvinit, debes completar la instalación de este paquete creando el enlace simbólico `/bin/pidof` ejecutando:

```
ln -s killall /bin/pidof
```

Instalación de Shadow-4.0.3

```
Estimación del tiempo de construcción:       0.88 SBU
Estimación del espacio necesario en disco:  7 MB
```

Contenido de Shadow

Última versión comprobada: 4.0.3.

El paquete Shadow se creó para consolidar la seguridad del sistema de contraseñas.

Shadow instala lo siguiente:

Programas

chage, chfn, chpasswd, chsh, dpasswd, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, groups, grpck, grpconv, grpunconv, lastlog, login, logoutd, mkpasswd, newgrp, newusers, passwd, pwck, pwconv, pwunconv, sg (enlace a newgrp), useradd, userdel, usermod, vigr (enlace a vipw) y vipw

Librerías

libmisc y libshadow

Dependencias de instalación de Shadow

Última versión comprobada: 20001016.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, nm, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gettext: msgfmt, xgettext
Gcc: cc1, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Texinfo: makeinfo
Textutils: cat, sort, tr, uniq

Instalación del Entorno de Contraseñas Ocultas (Shadow Password Suite)

Antes de instalar este paquete, puede que quieras echar un vistazo a la receta de Shadow. En ella se discute cómo puedes hacer más seguro tu sistema en lo relativo al uso de contraseñas, por ejemplo activando el uso de las contraseñas MD5, que son más seguras, o cómo conseguir exprimir hasta el máximo este paquete. Puedes encontrar la receta de Shadow en http://www.escomposlinux.org/lfs−es/recetas/shadowpasswd_plus.html (la versión original se encuentra en http://hints.linuxfromscratch.org/hints/shadowpasswd_plus.txt).

Prepara Shadow para su compilación:

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

Continúa compilando el paquete:

```
make
```

Instala el paquete:

```
make install
```

Shadow utiliza dos ficheros para configurar los ajustes de autenticación para el sistema. Instala estos ficheros de configuración:

```
cp etc/{limits,login.access} /etc
```

`/var/spool/mail` es la antigua localización de los buzones de los usuarios. El lugar que se usa hoy en día es /var/mail. Ejecuta el siguiente comando para cambiar la localización del buzón de correo:

```
sed 's%/var/spool/mail%/var/mail%' \
    etc/login.defs.linux > /etc/login.defs
```

De acuerdo a la página de manual de **`vipw`**, debería existir un enlace simbólico **`vigr`**. Como el procedimiento de instalación de shadow no crea este enlace simbólico, lo hacemos manualmente:

```
ln -s vipw /usr/sbin/vigr
```

El enlace `vipw` apunta actualmente a un fichero que no existe. Puesto que no necesitamos este fichero aquí, lo eliminamos

```
rm /bin/vipw
```

Mueve el programa **`sg`** al directorio `/usr/bin`:

```
mv /bin/sg /usr/bin
```

Nueve las librerías dinámicas de Shadow a un lugar más apropiado:

```
mv /usr/lib/lib{shadow,misc}.so.0* /lib
```

Las librerías has sido movidas, pero algunos paquetes espera encontrarlas en el directorio `/usr/lib`. Para solventar esto, crea los siguientes enlaces simbólicos:

```
ln -sf ../../lib/libshadow.so.0 /usr/lib/libshadow.so &&
ln -sf ../../lib/libmisc.so.0 /usr/lib/libmisc.so
```

Tanto Sh−utils como Shadow instalan un programa `groups` que es casi identico. Si lo deseas, puedes eliminar el programa `groups` instalado por Shadow ejecutando el siguiente comando:

```
rm /bin/groups
```

Configuración del entorno de contraseñas ocultas (Shadow Password Suite)

Este paquete contiene las utilidades para modificar las contraseñas de los usuarios, añadir o borrar nuevos usuarios y grupos, y similar. No vamos a explicar lo que significa 'password shadowing'. Puedes econtrar una completa explicación en el fichero doc/HOWTO que está en el árbol de fuentes de la Shadow Password Suite al desempaquetarla. Hay una cosa que debes recordar si decides usar soporte para contraseñas ocultas: los programas que necesiten verificar contraseñas (por ejemplo xdm, demonios de ftp, demonios de pop3, etc) necesitarán ser 'compatibles con shadow', es decir, necesitan ser capaces de trabajar con 'shadow passwords'.

Para habilitar las contraseñas ocultas, ejecuta el siguiente comando:

```
/usr/sbin/pwconv
```

Instalación de Sysklogd–1.4.1

```
Estimación del tiempo de construcción:       0.03 SBU
Estimación del espacio necesario en disco:  472 KB
```

Contenido de Sysklogd

Última versión comprobada: 1.4.1.

El paquete Sysklogd contiene programas para grabar tanto los mensajes del sistema como los generados por el núcleo

Sysklogd instala lo siguiente:

Programas

klogd y syslogd

Dependencias de instalación de Sysklogd

Última versión comprobada: 1.4.1.

Binutils: as, ld, strip
Fileutils: install
Gcc: cc1, collect2, cpp0, gcc
Make: make

Instalación de Sysklogd

Prepara y compila Sysklogd:

```
make
```

Y termina instalando el paquete:

```
make install
```

Configuración de Sysklogd

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

```
cat > /etc/syslog.conf << "EOF"
# Inicio de /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 *

# Fin de /etc/syslog.conf
EOF
```

Instalación de Sysvinit–2.84

```
Estimación del tiempo de construcción:       0.06 SBU
Estimación del espacio necesario en disco:  1 MB
```

Contenido de Sysvinit

Última versión comprobada: 2.84.

El paquete Sysvinit contiene programas para controlar el arranque, ejecución y descarga de todos los demás programas.

Sysvinit instala lo siguiente:

Programas

halt, init, killall5, last, lastb (enlace a last), mesg, pidof (enlace a killall5), poweroff (enlace a halt), reboot (enlace a halt), runlevel, shutdown, sulogin, telinit (enlace a init), utmpdump y wall

Dependencias de instalación de Sysvinit

Última versión comprobada: 2.84.

Bash: sh
Binutils: as, ld

Fileutils: chown, cp, install, ln, mknod, rm
Gcc: cc, cc1, collect2, cpp0
Make: make
Sed: sed

Instalación de Sysvinit

Cuando se cambia de nivel de ejecución (por ejemplo cuando apagamos el sistema) el programa init envía las señales TERM y KILL a todos los procesos que él mismo inició, mostrando por pantalla el mensaje "Sending processes the TERM signal" (Enviando la señal TERM a los procesos). Esto parece indicar que init envía esta señal a todos los procesos en ejecución. Para evitar esta confusión, puede modificarse el fichero init.c de manera que el mensaje diga "Sending processes started by init the TERM signal" (Enviando la señal TERM a los procesos iniciados por init), ejecutando los siguientes comandos. Si no quieres cambiarlo, sáltate este paso.

Edita el mensaje de parada:

```
cp src/init.c{,.backup} &&
sed 's/Sending processes/Sending processes started by init/g' \
    src/init.c.backup >
src/init.c
```

Compila Sysvinit:

```
make -C src
```

Y termina instalando el paquete:

```
make -C src install
```

Configuración de Sysvinit

Crea un nuevo fichero `/etc/inittab` ejecutando lo siguiente:

```
cat > /etc/inittab << "EOF"
# Inicio de /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

# Fin de /etc/inittab
EOF
```

Instalación de Tar–1.13

```
Estimación del tiempo de construcción:       0.26 SBU
Estimación del espacio necesario en disco:  6 MB
```

Contenido de Tar

Última versión comprobada: 1.13.

Tar es un programa de archivado diseñado para almacenar y estraer ficheros en un archivo conocido como fichero tar.

Tar instala lo siguiente:

Programas

rmt y tar

Dependencias de instalación de Tar

Última versión comprobada: 1.13.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk

Net−tools: hostname
Patch: patch
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Instalación de Tar

Se debe aplicar un parche para que tar tenga soporte directo para los ficheros bz2. Este parche añade la opción *−j* a tar, que es similar a la opción *−z* utilizada para los ficheros gzip.

Aplica el parche ejecutando el siguiente comando:

```
patch -Np1 -i ../tar-1.13.patch
```

Prepara Tar para su compilación:

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

Continúa compilando el paquete:

```
make
```

Y termina instalando el paquete:

```
make install
```

Instalación de Util−linux−2.11y

```
Estimación del tiempo de construcción:       0.38 SBU
Estimación del espacio necesario en disco:  10 MB
```

Contenido de Util−linux

Última versión comprobada: 2.11t.

El paquete Util−linux contiene una miscelánea de utilidades. Alguna de estas utilidades más destacables se utilizan para montar, desmontar, formatear, particionar y manejar dispositivos de disco, abrir puertos de consola o capturar los mensajes del núcleo.

Util−linux instala lo siguiente:

Programas

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, 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, parse.bash, parse.tcsh, pg, pivot_root, ramsize (enlace a rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (enlace a rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (enlace a swapon), swapon, test.bash, test.tcsh, tunelp, ul, umount, vidmode (enlace a rdev), whereis y write

Dependencias de instalación de Util−linux

Última versión comprobada: 2.11n.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chgrp, chmod, cp, install, ln, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp, cpp0
Glibc: rpcgen
Grep: grep
Make: make
Sed: sed
Sh−utils: uname, whoami
Textutils: cat

Notas sobre la conformidad con el estándar FHS

El estándar FHS recomienda que usemos /var/lib/hwclock para la ubicación del archivo adjtime, en lugar del habitual /etc. Para hacer que hwclock sea conforme a FHS, ejecuta lo siguiente:

```
cp hwclock/hwclock.c{,.backup} &&
sed 's%etc/adjtime%var/lib/hwclock/adjtime%' \
    hwclock/hwclock.c.backup > hwclock/hwclock.c &&
mkdir -p /var/lib/hwclock
```

Instalación de Util−linux

Prepara Util−linux para su compilación:

```
./configure
```

Continúa compilando el paquete:

```
make HAVE_SLN=yes
```

La opción *HAVE_SLN* evita que este paquete, que fué instalado por Glibc, sea construido otra vez.

Y termina instalando el paquete:

```
make HAVE_SLN=yes install
```

Instalación de las páginas de manual de Linux–2.4.20

```
Estimación del tiempo de construcción:       0.01 SBU
Estimación del espacio necesario en disco:  Por determinar
```

Contenido de Linux

Última versión comprobada: 2.4.18.

El núcleo Linux es el corazón de todo sistema Linux. Es lo que hace a Linux funcionar. Cuando se enciende un ordenador y se inicia un sistema Linux, el núcleo es lo primero que se carga. El núcleo inicializa los componentes hardware del sistema: puertos serie, puertos paralelo, tarjetas de sonido, tarjetas de red, controladores IDE, controladores SCSI y mucho más. En pocas palabras, el núcleo hace que el hardware esté disponible para que el software pueda ejecutarse.

Linux instala lo siguiente:

Programas

El núcleo y las cabeceras del núcleo

Instalación de las páginas de manual del núcleo

Construye las páginas de manual:

```
make mandocs
```

Instala las páginas de manual:

```
cp -a Documentation/man /usr/share/man/man9
```

Instalación de Glibc–2.3.1

```
Tiempo estimado de construcción:       Por determinar
Estimación del espacio necesario en disco:  Por determinar
```

Contenido de Glibc

Última versión comprobada: 2.2.5.

Glibc es la librería C que proporciona las llamadas al sistema y las funciones básicas, tales como open, malloc, printf, etc. La librería C es utilizada por todos los programas enlazados dinámicamente.

Glibc instala lo siguiente:

Programas

catchsegv, gencat, getconf, getent, glibcbug, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, nscd_nischeck, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump y zic

Librerías

ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd−compat.a, libc.[a,so], libc_nonshared.a, libcrypt.[a,so], libdl.[a,so], libg.a, libieee.a, libm.[a,so], libmcheck.a, libmemusage.so, libnsl.a, 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]

Dependencias de instalación de Glibc

Última versión comprobada: 2.2.5.

Bash: sh
Binutils: ar, as, ld, ranlib, readelf
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mknod, mv, mkdir, rm, touch
Gcc: cc, cc1, collect2, cpp, gcc
Grep: egrep, grep
Gzip: gzip
Make: make
Gawk: gawk
Sed: sed
Sh−utils: date, expr, hostname, pwd, uname
Texinfo: install−info, makeinfo
Textutils: cat, cut, sort, tr

Instalación de Glibc

Al principio de este capítulo instalaste Glibc aplicandole un parche. Parte de este parche deshaciá algunos cambios para hacer que los binarios compilados con Glibc−2.2 funcionasen. Sin embargo, esto no es lo que los desarrolladores de Glibc intentan y nosotros no necesitamos mantener esta Glibc modificada. Así que reinstalamos aquí Glibc para eliminar este parche.

Una segunda razón para reinstalar Glibc de nuevo es para poder considerarla límpia. La primera Glibc se instaló usando programas compilados en tu sistema anfitrión que, en ocasiones, tienen el efecto de corromper Glibc. Mientras que este no es un problema con los otros programas compilados anteriormente en este capítulo, para Glibc queremos asegurarnos de que es 100% correcta (de todas formas, eres libre de reinstalar otros paquetes en este punto para poder decir que has compilado un sistema LFS con LFS. Algo similar al metodo de instalación de autocompilación (bootstrap) de GCC.

También instalaremos aquí las páginas de manual de linuxthreads. Como puedes recordar, no fué posible hacerlo durante la primera instalación de Glibc debido a que aún no estaba instalado Perl. Ahora que todo lo que necesitamos para instalar las páginas de manual de linuxthreads está presente, las instalaremos también.

Antes de instalar Glibc, debes entrar al directorio glibc−2.3.1 y desempaquetar glibc−linuxthreads dentro del directorio glibc−2.3.1, no en /usr/src como normalmente harías.

Se sabe que este programa se comporta mal si cambias sus parámetros de optimización (incluyendo las opciones −march y −mcpu). Por tanto, si tienes definida cualquier variable de entorno que pueda sobreescribir las optimizaciones por defecto, como CFLAGS y CXXFLAGS, te recomendamos que las desactives o modifiques antes de construir Glibc.

Básicamente, compilar Glibc de forma diferente a como el libro sugiere pone tu sistema en grave riesgo.

La documentación sobre la instalación de Glibc recomienda construir Gluibc fuera del árbol de las fuentes. Crea el directorio de construcción:

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

A continuación, prepara Glibc para su compilación:

```
../glibc-2.3.1/configure --prefix=/usr --disable-profile \
    --enable-add-ons --libexecdir=/usr/bin
```

Continúa compilando el paquete:

```
make
```

Inicia la instalación del paquete:

```
make install
```

Construye las páginas de manual de linuxthreads:

```
make -C ../glibc-2.3.1/linuxthreads/man
```

Instala las páginas de manual:

```
make -C ../glibc-2.3.1/linuxthreads/man install
```

Completa la instalación del paquete recargando bash:

```
exec /bin/bash --login
```

Comando chroot revisado

A partir de ahora, cuando salgas del entorno chroot y desees reentrar en él necesitarás ejecutar el siguiente comando chroot modificado. El que se utiliza al principio de este capítulo puede que no vuelva a funcionar (si tu distribución anfitriona está basada en glibc−2.2.x o anterior, los programas en `/static/bin`, como

bash, no funcionarán). El siguiente comando chroot funcionará independientemente de la versión de Glibc de tu distribución anfitriona.

Adicionalmente, ahora que todo el software ha sido instalado, no hay necesidad de utilizar nada del directorio /static.

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

Instalación de LFS-Bootscripts-1.11

```
Estimación del tiempo de construcción:       0.01 SBU
Estimación del espacio necesario en disco:  252 KB
```

Contenido de LFS-bootscripts

Última versión comprobada: 1.11.

El paquete LFS-Bootscripts contiene guiones del interprete de comandos para el inicio del sistema al estilo SysV. Estos guiones realizan varias tareas como comprobar la integridad del sistema de ficheros durante el arranque, cargar el mapa del teclado, establecer la red y parar los procesos durante el cierre del sistema.

LFS-bootscripts instala lo siguiente:

Guiones

checkfs, cleanfs, functions, halt, ifdown, ifup, loadkeys, localnet, mountfs, mountproc, network, rc, reboot, sendsignals, setclock, swap, sysklogd y template

Dependencias de instalación de LFS-Bootscripts

Última versión comprobada: 1.11.

Fileutils: chown, cp

Instalación de LFS-Bootscripts

Nosotros usamos guiones de inicio al estilo SysV. Lo hemos elegido porque es ampliamente usado y nos sentimos cómodos con él. Si quieres probar alguna otra cosa, Marc Heerdink ha escrito una receta sobre los guiones de arranque al estilo BSD, que puedes encontrar en http://www.escomposlinux.org/lfs-es/recetas/bsd-init.html (la versión original se encuentra en http://hints.linuxfromscratch.org/hints/bsd-init.txt).

Si decides usar el estilo BSD o cualquier otro estilo de guiones, puedes saltarte el Capítulo 7 e ir directamente al Capítulo 8.

Instala los guiones de arranque:

```
cp -a rc.d sysconfig /etc
```

Asignale a *root* la propiedad de los guiones:

```
chown -R root:root /etc/rc.d /etc/sysconfig
```

Configuración de los componentes del sistema

Ahora que están todos los paquetes instalados, lo que necesitamos hacer son algunas tareas de configuración.

Configuración del teclado

Nada es más molesto que usar Linux teniendo cargado un mapa de teclado incorrecto. Si tienes un teclado estándar de US (EEUU), te puedes saltar esta sección. El mapa de teclado US es el mapa por defecto si no lo cambias.

Para asignar un mapa de teclado por defecto, crea el enlace simbólico `/usr/share/kbd/keymaps/defkeymap.map.gz` ejecutando los siguientes comandos:

```
ln -s <ruta/al/mapa/del/teclado> /usr/share/kbd/keymaps/defkeymap.map.gz
```

Reemplaza <ruta/al/mapa/del/teclado> por tu fichero de mapa de teclado. Por ejemplo, si tienes un teclado español, deberías ejecutar:

```
ln -s i386/qwerty/es.map.gz defkeymap.map.gz
```

La segunda opción para configurar la disposición de tu teclado es compilar el mapa de teclado directamente en el núcleo. Esto asegurará que tu teclado siempre funcione como se espera, incluso cuando has arrancado en modo de rescate (pasando `init=/bin/sh' al núcleo) y los guiones de arranque que normalmente se encargan de cargar el mapa de teclado adecuado no se hayan ejecutado.

Ejecuta el siguiente comando para parchear el mapa de teclado correcto dentro de las fuentes del núcleo. Debes repetir este comando siempre que desenpaquetes un nuevo núcleo:

```
loadkeys -m /usr/share/kbd/keymaps/defkeymap.map.gz > \
    /usr/src/linux-2.4.20/drivers/char/defkeymap.c
```

Creación de los ficheros /var/run/utmp, /var/log/wtmp y /var/log/btmp

Programas como login, shutdown, uptime y otros necesitan leer y escribir en /var/run/utmp, /var/log/btmp y /var/log/wtmp. Estos ficheros contienen información acerca de quién está conectado en ese momento. También contienen información acerca de cuándo fue arrancado y parado por última vez el ordenador y un registro de los intentos de conexión fallidos.

Crea estos ficheros con los permisos apropiados ejecutando los siguientes comandos:

```
touch /var/run/utmp /var/log/{btmp,lastlog,wtmp} &&
chmod 644 /var/run/utmp /var/log/{btmp,lastlog,wtmp}
```

Creación de la contraseña de root

Elige una contraseña para el usuario administrador (root) y créala ejecutando el siguiente comando:

```
passwd root
```

Capítulo 7. Preparando los guiones de arranque

Introducción

En este capítulo se configurarán los guiones de arranque que has instalado en el capítulo 6. Muchos de estos guiones funcionarán sin necesidad de modificaciones, pero algunos requieren ficheros de configuración adicionales para que puedan manejar la información dependiente del hardware específico de tu sistema.

¿Cómo hacen estos guiones que funcione el proceso 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 <inserte el nombre de una distribución> 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 (desde el 0 al 6) niveles de ejecución (en realidad, existen más pero son para casos especiales y es raro utilizarlos. Puedes leer la página man de init para obtener más información), 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 (como 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 deseemos 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. En el momento que se desee cambiar de nivel se pararán los servicios del nivel actual, para iniciar los del nuevo nivel.

Los enlaces simbólicos apuntan a los guiones situados en el directorio /etc/rc.d/init.d, que son los que realmente se ejecutan. Tanto los enlaces de parada como los de inicio apuntan al mismo guión. Esto se debe a que se pueden ejecutar usando parámetros como start, stop, restart, reload o status. Cuando se encuentra un enlace que comienza por K se ejecuta el guión con el parámetro stop. Y cuando comienza por S, con el parámetro start.

Hay una excepción. Los enlaces que comienzan por S en los directorios rc0.d y rc6.d no inician nada. Todos estos guiones se ejecutan con el parámetro *stop* para parar algo. Es evidente que cuando quieres apagar o reiniciar el sistema, no quieres ejecutar nada, sólo quieres pararlo.

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 se necesita reiniciar el servicio para que actualice su configuración.
- *status*: Dice si el servicio se está ejecutando y con qué identificador de proceso (PID).

Por supuesto, puedes modificar el proceso de inicio para adecuarlo a tus necesidades (después de todo es tu sistema LFS). Lo aquí expuesto es tan sólo un ejemplo de cómo hacer las cosas de una manera correcta (claro que aunque a nosotros esta manera nos parezca bien, puede que tú la odies).

Configuración del guión setclock

El guión setclock lee la hora del reloj interno del ordenador (también conocido como el reloj CMOS o BIOS) y la convierte a la hora local mediante el fichero `/etc/localtime` (si el reloj interno del ordenador utiliza GMT) o no (si el reloj interno de la computadora ya está puesto a la hora local). No hay manera de detectar automáticamente si el reloj utiliza GMT o no, así que necesitamos configurarlo nosotros mismos.

Si el reloj interno del ordenador no utiliza GMT hay que cambiar el valor de la variable *UTC* a *0* (cero).

Para ello vamos a crear el fichero `/etc/sysconfig/clock` mediante la ejecución del siguiente comando:

```
cat > /etc/sysconfig/clock << "EOF"
# Inicio de /etc/sysconfig/clock

UTC=1

# Fin de /etc/sysconfig/clock
EOF
```

Para más información sobre la hora en LFS tienes una buena explicación en http://www.escomposlinux.org/lfs−es/recetas/time.html (la versión original se encuentra en http://hints.linuxfromscratch.org/hints/time.txt). En él se explican conceptos como las zonas horarias, UTC, y la variable de entorno TZ.

¿Necesito el guión loadkeys?

Si decidiste compilar tu fichero de mapa de caracteres dentro del núcleo al final del Capítulo 6, no es estrictamente necesario ejecutar el guión loadkeys, ya que será el núcleo el que activará dicho mapa. Aunque puedes ejecutarlo de todas maneras sin que te cause ningún problema. De todas formas, es beneficioso que mantengas el guión en el caso de tener varios núcleos y no te acuerdes o no quieras introducir el fichero de mapa de caracteres en todos ellos.

Si has decidido que no necesitas o que no quieres el guión loadkeys, elimina el enlace simbólico `/etc/rc.d/rcsysinit.d/S70loadkeys`

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 se escribe por defecto en el fichero de registro cada 20 minutos. Si quieres habilitar esta marca de tiempo periódica debes editar el guión `sysklogd` y realizar los cambios necesarios. Para más información utiliza el comando **`man syslogd`**.

Configuración del guión localnet

Una de las cosas que hace el guión localnet es establecer el nombre de la máquina. Es necesario configurar dicho nombre en /etc/sysconfig/network.

Puedes crear el fichero /etc/sysconfig/network y configurar el nombre de tu máquina ejecutando:

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

Debes substituir "lfs" por el nombre de tu máquina. No debes escribir el FQDN (nombre completo de la máquina, incluido su dominio). Esta información la escribiremos más tarde en el fichero `/etc/hosts`

Creación del fichero /etc/hosts

Si se va a configurar una tarjeta de red, debes decidir la dirección IP, el FQDN y los posibles alias para escribirlos en el fichero /etc/hosts. La sintaxis es:

```
<dirección IP> minombre.midominio.org alias
```

Debes asegurarte de utilizar una dirección IP que pertenezca al rango de direcciones IP privadas. Los rangos válidos son:

```
        Clases de redes
        A     10.0.0.0
        B     Entre 172.16.0.0 y 172.31.0.0
        C     Entre 192.168.0.0 y 192.168.255.0
```

Una dirección IP válida puede ser 192.168.1.1. Un FQDN válido para esa dirección IP puede ser www.linuxfromscratch.org.

Aunque no vayas a configurar la tarjeta de red necesitas un FQDN, ya que algunos programas lo necesitan para funcionar correctamente.

Si no vas a configurar la tarjeta de red crea el fichero `/etc/hosts` ejecutando:

```
cat > /etc/hosts << "EOF"
# Inicio de /etc/hosts (versión sin tarjeta de red)

127.0.0.1 www.midominio.com <valor de HOSTNAME> localhost

# Fin de /etc/hosts (versión sin tarjeta de red)
```

```
EOF
```

Si vas a configurar la tarjeta de red crea el fichero `/etc/hosts` ejecutando:

```
cat > /etc/hosts << "EOF"
# Inicio de /etc/hosts (versión con tarjeta de red)

127.0.0.1 localhost.localdomain localhost
192.168.1.1 www.midominio.org <valor de HOSTNAME>

# Fin de /etc/hosts (versión con tarjeta de red)
EOF
```

Por supuesto, puedes cambiar 192.168.1.1 y www.midominio.org a tu gusto (o a lo que te indique el administrador de la red/sistema si está planeado que se conecte esta máquina a una red que ya existe).

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, debes eliminar los enlaces simbólicos a `network` de todos los directorios de los niveles de ejecución (`/etc/rc.d/rc*.d`)

Configuración de la puerta de enlace por defecto

Si estás conectado a una red puede que necesites establecer cual es la puerta de enlace por defecto para esa máquina. Para ello, se deben añadir los valores apropiados al fichero /etc/sysconfig/network ejecutando lo siguiente:

```
cat >> /etc/sysconfig/network << "EOF"
GATEWAY=192.168.1.2
GATEWAY_IF=eth0
EOF
```

Debes cambiar los valores de GATEWAY y GATEWAY_IF por los que correspondan en tu red. GATEWAY contiene la dirección IP de la puerta de enlace por omisión, y GATEWAY_IF la interfaz de red por la que es accesible dicha dirección IP.

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 situados en el directorio /etc/sysconfig/network–devices. Este directorio debe contener ficheros con el nombre ifconfig.xyz, donde xyz corresponde con el nombre de la interfaz de red (como eth0 o eth0:1).

Si decides renombrar o mover el directorio /etc/sysconfig/network–devices, asegúrate de que actualizas el fichero /etc/sysconfig/rc, asignando a la variable network_devices la nueva localización.

Ahora, los nuevos ficheros que creemos en este directorio contienen lo siguiente. Como ejemplo vamos a crear el fichero ifconfig.eth0 ejecutando:

```
cat > /etc/sysconfig/network-devices/ifconfig.eth0 << "EOF"
ONBOOT=yes
IP=192.168.1.1
NETMASK=255.255.255.0
BROADCAST=192.168.1.255
EOF
```

Por supuesto, los valores de estas variables se deben cambiar en todos los ficheros que creemos por los valores apropiados para nuestra máquina. Si la variable ONBOOT tiene el valor yes, el guión network activará la interfaz durante el arranque del sistema. Si contiene cualquier otro valor, el guión network ignorará el contenido del archivo y, por lo tanto, no la activará.

Capítulo 8. Hacer el sistema LFS arrancable

Introducción

Este capítulo hará arrancable el sistema LFS. Trataremos la creación de un nuevo fichero fstab, la construcción de un nuevo núcleo para el nuevo sistema LFS y la adición de las entradas apropiadas a LILO para que el sistema LFS se pueda seleccionar en la línea de comandos de LILO.

Creación del fichero /etc/fstab

Para que ciertos programas sean capaces de determinar dónde se supone que están montadas por defecto las particiones, se usa el fichero /etc/fstab. Crea un nuevo fichero `/etc/fstab` conteniendo lo siguiente:

```
cat > /etc/fstab << "EOF"
# Inicio de /etc/fstab

# sistema de   punto de      tipo del       opciones  volcado  orden de
# archivos       montaje     sistema de                                 chequeo
#                                       archivos

/dev/*LFS*       /                 *tipo*           defaults       1            1
/dev/*swap*     swap           swap          pri=1           0            0
proc                 /proc           proc          defaults        0            0

# Fin de /etc/fstab
EOF
```

LFS, ***swap*** y ***tipo*** deben ser reemplazados por los valores apropiados (/dev/hda2, /dev/hda5 y reiserfs, por ejemplo).

Cuando se añada una partición reiserfs, los valores **1 1** que aparecen al final de la línea deberían cambiarse a **0 0**.

Para más información sobre los campos que aparecen en el fichero fstab, ver **man 5 fstab**.

Existen otras líneas que puedes considerar añadir al fichero fstab. Un ejemplo es la línea que debería contener si estás utilizando devpts:

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

Otro ejemplo es la línea a incluir si pretendes utilizar dispositivos USB:

```
usbfs       /proc/bus/usb usbfs defaults    0       0
```

Tanto una como otra opción sólo funcionarán si se tiene el soporte pertinente compilado dentro del núcleo.

Instalación de Linux–2.4.20

```
Estimación del tiempo de construcción:      Todas las opciones por defecto: 4.20 SBU
Estimación del espacio necesario en disco:  Todas las opciones por defecto: 181 MB
```

Contenido de Linux

Última versión comprobada: 2.4.18.

El núcleo Linux es el corazón de todo sistema Linux. Es lo que hace a Linux funcionar. Cuando se enciende un ordenador y se inicia un sistema Linux, el núcleo es lo primero que se carga. El núcleo inicializa los componentes hardware del sistema: puertos serie, puertos paralelo, tarjetas de sonido, tarjetas de red, controladores IDE, controladores SCSI y mucho más. En pocas palabras, el núcleo hace que el hardware esté disponible para que el software pueda ejecutarse.

Linux instala lo siguiente:

Programas

El núcleo y las cabeceras del núcleo

Dependencias de instalación de Linux

Última versión comprobada: 2.4.17.

Bash: sh
Binutils: ar, as, ld, nm, objcopy
Fileutils: cp, ln, mkdir, mv, rm, touch
Findutils: find, xargs
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Gzip: gzip
Make: make
Gawk: awk
Modutils: depmod, genksyms
Net−tools: dnsdomainname, hostname
Sed: sed
Sh−utils: basename, date, expr, pwd, stty, uname, whoami, yes
Textutils: cat, md5sum, sort, tail

Instalación del núcleo

Construir el núcleo comprende dos pasos: configurarlo y compilarlo. Hay varias maneras de configurar el núcleo. Si no te gusta la que utiliza este libro, lee el fichero `README` que acompaña al árbol de código fuente del núcleo, y busca qué otras opciones existen.

Prepara la compilación ejecutando el siguiente comando:

```
make mrproper
```

Esto asegura que las fuentes del núcleo están completamente limpias. 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 despues de desempaquetarlo.

Configura el núcleo mediente una interfaz de menús:

```
make menuconfig
```

Puede que **`make oldconfig`** sea mejor elección en algunas situaciones. Lee el fichero `README` para más detalles.

Si lo deseas, puedes saltarte la configuración del núcleo copiando el fichero de configuración del núcleo, `.config`, de tu sistema anfitrión al directorio `$LFS/usr/src/linux-2.4.20`.

Comprueba las dependencias y crea los ficheros de información de las dependencias:

```
make dep
```

Compila la imagen del núcleo:

```
make bzImage
```

Compila los controladores que han sido configurados como módulos:

```
make modules
```

Si intentas usar los módulos del núcleo necesitas el fichero `/etc/modules.conf`. La información relativa a los módulos, y a la configuración del núcleo en general, puedes encontrarla en la documentación del núcleo, que se guarda en `/usr/src/linux-2.4.20/Documentation`. La página de manual de modules.conf y el kernel–CÓMO en http://es.tldp.org/COMO–INSFLUG/COMOs/Kernel–Como/ (el original se encuentra en http://www.tldp.org/HOWTO/Kernel–HOWTO.html) puede que también sean de interes para ti.

Instala los módulos:

```
make modules_install
```

La compilación del núcleo ha terminado, pero algunos de los ficheros creados aún residen en el árbol de las fuentes. Para completar la instalación, dos ficheros deben copiarse al directorio `/boot`.

La ruta al fichero del núcleo puede variar dependiendo de la plataforma que utilices. Ejecuta el siguiente comando para instalar el núcleo:

```
cp arch/i386/boot/bzImage /boot/lfskernel
```

`System.map` es un fichero de símbolos para el núcleo. Mapea los puntos de entrada de las funciones de cada función 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 del mapa:

```
cp System.map /boot
```

Hacer el sistema LFS arrancable

Para poder arrancar el sistema LFS, necesitamos actualizar nuestro gestor de arranque. Asumiremos que tu sistema original usa Lilo (ya que es el gestor de arranque más comúnmente usado en este momento).

No ejecutaremos el programa lilo dentro del entorno chroot. Ejecutar lilo dentro de un entorno chroot puede acarrear fatales efectos secundarios que lleguen a inutilizar tu Registro Maestro de Arranque (MBR – Master Boot Record), y necesitarías un disco de arranque para poder arrancar cualquier sistema Linux (tanto el original como el sistema LFS).

Primero, saldremos del entorno chroot y copiaremos el fichero lfskernel al sistema original:

```
logout
cp $LFS/boot/lfskernel /boot
```

El siguiente paso es añadir una entrada en /etc/lilo.conf, para que podamos escoger a LFS cuando reiniciemos el ordenador:

```
cat >> /etc/lilo.conf << "EOF"
image=/boot/lfskernel
        label=lfs
        root=<partición>
        read-only
EOF
```

<partición> debe ser reemplazada con el nombre de la partición LFS.

Ten en cuenta que si estás utilizando el sistema de ficheros reiserfs para tu partición raíz, se debe cambiar la línea **`read-only`** por **`read-write`**.

Ahora, actualiza el gestor de arranque ejecutando:

```
/sbin/lilo -v
```

El último paso es sincronizar los ficheros de configuración de lilo del sistema original con los del sistema LFS:

```
cp /etc/lilo.conf $LFS/etc &&
cp $(grep "image.*=" /etc/lilo.conf | cut -f 2 -d "=") $LFS/boot
```

Capítulo 9. El final

El final

¡Bien hecho! Has terminado de instalar tu sistema LFS. Puede que haya sido un proceso largo pero esperamos que haya merecido la pena. Te deseamos mucha diversión con tu flamante sistema Linux hecho a la medida.

Ahora podría ser un buen momento para quitar todos los símbolos de depurado de los archivos binarios de tu sistema LFS. Si no eres un programador y no planeas depurar tus programas, entonces te alegrará saber que puedes recuperar algunas decenas de megabytes borrando estos símbolos. Este proceso no produce ningún otro inconveniente que no sea no poder depurar los programas nunca más, lo que no es problema si no sabes cómo depurarlos.

Advertencia: El 98% de la gente que usa el comando mencionado más adelante no experimenta ningún problema. Pero haz una copia de seguridad de tu sistema LFS antes de ejecutar este comando. Hay una pequeña posibilidad de que te salga el tiro por la culata, y convierta tu sistema en inutilizable (principalmente destruyendo los módulos del núcleo y las librerías dinámicas y compartidas). Sin embargo, suele ocurrir más a menudo por un error tipográfico que por un problema con el comando utilizado.

Después de haber dicho esto, la opción −−strip−debug que usaremos para quitar los símbolos de depuración es, bajo circunstancias normales, bastante inocua. No borrará nada vital de los ficheros. También es bastante seguro usar −−strip−all con programas normales (no se debe usar en librerías −se destruirían), pero no es tan seguro como el anterior y el espacio que ganas no es tan grande. Pero si andas justo de espacio de disco, cada granito de arena ayuda, así que decide por ti mismo. Por favor, lee la página del manual (man) de strip para ver las opciones que puedes usar. La idea general es no ejecutar strip sobre librerías (usando otra opción que no sea −−strip−debug) para asegurarnos de hacer la apuesta segura.

```
find $LFS/{,usr/,usr/local/}{bin,sbin,lib} -type f \
   -exec /usr/bin/strip --strip-debug '{}' ';'
```

Puede ser una buena idea crear el fichero $LFS/etc/lfs. 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 el fichero $LFS/etc/lfs ejecutando el siguiente comando:

```
echo 4.1 > $LFS/etc/lfs
```

Registrarse

¿Quieres registrarte como usuario de LFS ahora que has terminado el libro? Visita http://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 el sistema LFS ahora...

Arranque del sistema

Ahora que se han instalado todos los programas , ya es hora de reiniciar el ordenador. Antes de reiniciar, desmontemos $LFS/proc y la partición LFS, ejecutando

```
umount $LFS/proc &&
umount $LFS
```

Si has decidido crear varias particiones, necesitas desmontar las otras particiones antes de desmontar $LFS, por ejemplo:

```
umount $LFS/proc &&
umount $LFS/usr &&
umount $LFS/home &&
umount $LFS
```

Y ahora puedes reiniciar el sistema ejecutando algo como:

```
/sbin/shutdown -r now
```

Asegúrate de indicar, en la línea de órdenes de LILO, que arranque *lfs* y no la entrada por defecto que haría arrancar de nuevo tu sistema original.

Una vez hayas reiniciado, tu sistema LFS está listo para su uso, y puedes empezar a añadir los programas que desees.

Una última cosa que puede que quieras hacer es ejecutar lilo, ahora que ya estás dentro del sistema LFS. De esta forma, pondrás en el MBR la versión de LILO del sistema LFS, en lugar de la que hay allí en este momento, que es la de tu sistema original. Dependiendo de cómo sea de antigua tu distribución original, la versión del sistema LFS puede tener más características avanzadas que necesites/puedas usar.

De cualquier forma, ejecuta lo siguiente para activar la versión de lilo instalada en el sistema LFS:

```
/sbin/lilo
```

Ahora puedes eliminar el directorio static. Si piensas que puedes necesitar rehacer el Capítulo 5, entonces querrás guardar una copia de respaldo del directorio antes de eliminarlo. Para eliminar el directorio static, escribe el siguiente comando:

```
rm -rf /static
```

Y ahora, ¿qué?

Te agradecemos que hayas leido el Libro LFS y esperamos que lo hayas encontrado útil y te recompense el tiempo empleado.

Ahora que has terminado de instalar tu sistema LFS, puede que te preguntes "Y ahora, ¿qué?". Para responder esta cuestión, te hemos preparado una lista de recursos.

- Más Allá de Linux From Scratch

  El libro Más Allá de Linux From Scratch cubre los procesos de instalación de una amplia gama de software que está más allá del alcance del Libro LFS. Puedes encontrar el proyecto BLFS en http://beyond.linuxfromscratch.org/, y su traducción al castellano en http://www.escomposlinux.org/lfs-es/blfs-es-CVS/.
- Recetas de LFS

Las Recetas de LFS son una serie de pequeños documentos educacionales suministrados por voluntarios a la comunidad LFS. Las Recetas están disponibles en http://hints.linuxfromscratch.org/hints.shtml. En http://www.escomposlinux.org/lfs−es/recetas/ puedes encontrar la traducción de un buén número de ellas.

- Listas de Correo

  Hay varias listas de correo sobre LFS a las que puedes suscribirte si necesitas ayuda. Mira el Capítulo 1 − Listas de correo y archivos para mas información.

  La comunidad hispanoparlante puede recurrir a la lista de correo linux−desde−cero. Esta lista de correo no pertenece oficialmente al proyecto LFS, pero igualmente encontrarás gente dispuesta a ayudarte.

  Si estás interesado en colaborar en la traducción al castellano de los diversos documentos del LFS, tienes a tu disposición la lista de correo lfs−es.

- 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. El sitio principal se encuentra en http://www.tldp.org/, y la sección en castellano en http://es.tldp.org.

IV. Parte IV – Apéndices

Índice

A. Descripción de paquetes y dependencias

Apéndice A. Descripción de paquetes y dependencias

Introducción

En este apéndice se describen los siguiente aspectos de cada paquete instalado en este libro.

- la localización oficial para la descarga del paquete.
- el contenido de cada paquete.
- lo que cada programa de dicho paquete hace.
- lo que cada paquete necesita para poder compilarlo.

Mucha de la información sobre estos paquetes (especialmente, su descripción) se ha extraído de las páginas del manual de esos paquetes. No incluimos las páginas del manual completas, sólo los elementos clave que hagan posible entender lo que cada programa hace. Para conseguir información detallada de un programa, dirijete a su página de manual o a su página info.

Ciertos paquetes están documentados con mayor profundidad que otros, sencillamente porque sabemos más sobre unos que sobre otros. Si algo debería ser añadido a las siguientes descripciones, por favor no dudes en comunicarlo en las listas de correo. Intentamos que la lista contenga una descripción detallada de cada paquete, pero no podemos hacerlo sin ayuda.

Ten en cuenta que actualmente sólo está descrito lo que hace un paquete , y no lo que necesita que esté instalado. Esto se añadirá más adelante.

También están listadas todas las dependencias para la instalación de todos los paquetes instalados en el libro. La lista incluye qué programas de qué paquetes son necesarios para compilar correctamente el paquete a instalar.

Estas no son las dependencias necesarias para su ejecución, por lo tanto no te ayudarán para saber qué programas son necesarios para usar los programas del paquete. Son solamente las dependencias necesarias para compilarlo.

La lista de dependencias puede estar en ocasiones anticuada con respecto a la versión del paquete usada actualmente. Comprobar las dependencias es un trabajo pesado por lo que puede haber un desfase en la actualización de los paquetes. Pero, normalmente, en la actualización de versiones menores del paquete, las dependencias de instalación no cambian, por lo que son actuales en muchos casos. Cuando actualizamos a una versión mayor, nos aseguramos de hacer también un chequeo de las dependencias.

Autoconf

Localización oficial para descarga

Autoconf (2.57):
ftp://ftp.gnu.org/gnu/autoconf/

Contenido de Autoconf

Última versión comprobada: 2.53.

Autoconf produce guiones del interprete de comandos que configuran automáticamente el código fuente.

Autoconf instala lo siguiente:

Programas

autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames

Descripciones

Última versión comprobada: 2.53.

Descripción de los programas

autoconf

autoconf es una herramienta para generar guiones del intérprete de comandos que automáticamente configuran paquetes de código fuente, adaptándolos a muchas clases de sistemas tipo UNIX. Los guiones de configuración creados por autoconf son independientes de autoconf cuando se ejecutan, por tanto sus usuarios no necesitan tenerlo instalado.

autoheader

El programa autoheader puede crear un plantilla de declaraciones #define de C, usada posteriormente por el guión configure.

autom4te

autom4te ejecuta GNU M4 sobre ficheros.

autoreconf

Si hay que generar varios guiones de configuración con autoconf, el programa autoreconf puede ahorrar algo de trabajo. Ejecuta autoconf y autoheader (cuando es necesario) repetidamente para recrear los guiones de configuración de autoconf y las plantillas de configuración de las cabeceras en el árbol de directorios actual.

autoscan

El programa autoscan ayuda en la creación de ficheros configure.in para los paquetes. Este programa analiza los ficheros fuente en el árbol de directorios. Si no se le especifica un directorio en la línea de comandos, utiliza el directorio de trabajo actual. Busca en los ficheros fuente problemas comunes de portabilidad y crea un fichero configure.scan que sirve como versión preliminar del fichero configure.in para ese paquete.

autoupdate

El programa autoupdate actualiza las llamadas a los macros de autoconf en los ficheros configure.in cambiando los nombres antiguos por los actuales.

ifnames

ifnames ayuda en la creación de los ficheros configure.in. Escribe los identificadores que el paquete usa en las construcciones condicionales del preprocesador de C. Si un paquete está preparado para tener cierta portabilidad, este programa le ayuda a averiguar lo que configure necesita comprobar. Puede ayudar a fijar ciertas carencias en los configure.in generados por autoscan.

Dependencias de instalación de Autoconf

Última versión comprobada: 2.52.

Bash: sh
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Grep: fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, tr

Automake

Localización oficial para descarga

Automake (1.7.2):
ftp://ftp.gnu.org/gnu/automake/

Contenido de Automake

Última versión comprobada: 1.6.2.

Automake genera ficheros Makefile.in, pensados para usar con Autoconf.

Automake instala lo siguiente:

Programas

acinstall, aclocal, aclocal−1.6, automake, automake−1.6, compile, config.guess, config.sub, depcomp, elisp−comp, install−sh, mdate−sh, missing, mkinstalldirs, py−compile, ylwrap

Descripciones

Última versión comprobada: 1.6.2.

Descripción de los programas

acinstall

acinstall es un guión que instala ficheros M4 del estilo aclocal.

aclocal, aclocal−1.6

automake incluye una serie de macros de autoconf que pueden ser usadas en los paquetes; algunas de ellas son requeridas por automake en ciertas situaciones. Estas macros deben estar definidas en el fichero aclocal.m4 o no serán vistas por autoconf.

El programa aclocal genera automáticamente los ficheros aclocal.m4 basados en el contenido de configure.in. Esto proporciona una forma conveniente de obtener las macros facilitadas por automake sin tener que buscarlas. Por otra parte, el mecanismo de aclocal es extensible para poder ser usado por otros paquetes

automake, automake−1.6

Para crear todos los Makefile.in de un paquete, ejecuta el programa automake en el directorio base, sin argumentos. automake automáticamente encontrará cada fichero Makefile.am apropiado (tras explorar configure.in) y generará el correspondiente Makefile.in.

compile

compile es un guión que actúa de envoltura (wrapper) para compiladores.

config.guess

config.guess es un guión que intenta averiguar el nombre canónico del sistema.

config.sub

config.sub es un guión con subrutinas para la validación de configuraciones.

depcomp

depcomp es un guión que compila un programa mientras genera dependencias como efecto lateral.

elisp−comp

elisp−comp es un guión que compila en octetos ficheros .el.

install−sh

install−sh es un guión que instala un programa, guión o fichero de datos.

mdate−sh

mdate−sh es un guión que imprime la fecha de modificación de un fichero o directorio.

missing

missing es un guión que actúa como sustituto común para varios programas GNU no encontrados durante una instalación.

mkinstalldirs

mkinstalldirs es un guión que genera una jerarquía de directorios.

py−compile

py−compile es un guión que compila un programa Python.

ylwrap

ylwrap es un guión que actúa como envoltorio en las invocaciones a lex/yacc.

Dependencias de instalación de Automake

Última versión comprobada: 1.5.

Bash: sh
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm, rmdir
Grep: fgrep, grep
Make: make
Perl: perl
Sed: sed
Sh−utils: echo, expr, hostname, sleep
Texinfo: install−info
Textutils: cat, tr

Bash

Localización oficial para descarga

Bash (2.05a):
ftp://ftp.gnu.org/gnu/bash/

Contenido de Bash

Última versión comprobada: 2.05a.

bash es la "Bourne−Again SHell", que es un completo intérprete de comandos usado ampliamente en sistemas Unix. El programa bash lee de la entrada estándar (el teclado). Un usuario escribe algo y el programa evalúa lo que ha escrito y hace algo con ello, como lanzar un programa.

Bash instala lo siguiente:

Programas

bash, sh (enlace a bash) y bashbug

Descripciones

Última versión conprobada: 2.05a.

Descripción de los programas

bash

bash es la "Bourne−Again SHell", que es un completo intérprete de comandos usado ampliamente en sistemas Unix. El programa bash lee de la entrada estándar (el teclado). Un usuario escribe algo y el programa evalúa lo que ha escrito y hace algo con ello, como lanzar un programa.

bashbug

bashbug es un guión que ayuda al usuario en la composición y envío de informes de errores relacionados con bash, en un formato estándar.

sh

sh es un 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.

Dependencias de instalación de Bash

Última versión comprobada: 2.05a.

Bash: bash, sh
Binutils: ar, as, ld, ranlib, size
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Gawk: awk

Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, tr, uniq

Bin86

Localización oficial para descarga

Bin86 (0.16.3):
http://www.cix.co.uk/~mayday/

Contenido de Bin86

Última versión comprobada: 0.16.3

Bin86 es un simple ensamblador y enlazador para código de máquinas 8086 − 80386.

Bin86 instala lo siguiente:

Programas

as86, as86_encap, ld86, nm86 (enlace a objdump86), objdump86 y size86 (enlace a objdump86)

Descripciones

Última versión comprobada: 0.16.3

Descripción de los programas

as86

as86 es un ensamblador para los procesadores 8086...80386.

as86_encap

as86_encap es un guión del intérprete de comandos que llama a as86 y que convierte el binario creado en un fichero prog.v de C para ser incluido o enlazado con programas como instaladores de bloques de arranque.

ld86

ld86 entiende solamente los ficheros objeto creados por el ensamblador as86. Puede enlazarlos tanto en un ejecutable I&D impuro como en uno separado.

nm86

La tabla de símbolos de un fichero binario.

objdump86

Vuelca información detallada sobre un fichero binario.

size86

Sumario de los tamaños y fechas de un fichero binario.

Dependencias de instalación de Bin86

Última versión comprobada: 0.16.0.

Bash: sh
Binutils: as, ld, strip
Fileutils: chmod, install, ln, mv
Gcc: cc, cc1, collect2, cpp0
Make: make
Sed: sed

Binutils

Localización oficial para descarga

Binutils (2.13.2):
ftp://ftp.gnu.org/gnu/binutils/

Contenido de Binutils

Última versión comprobada: 2.12.1.

Binutils es una colección de herramientas para el desarrollo de software que contiene un enlazador, un ensamblador y otras utilidades para trabajar con ficheros de objetos y archivos.

Binutils instala lo siguiente:

Programas

addr2line, ar, as, gprof, ld, nm, objcopy, objdump, ranlib, readelf, size, strings y strip

Librerías

libbfd.[a,so] y libopcodes.[a,so]

Descripciones

Última versión comprobada: 2.12.1.

Descripción de los programas

addr2line

addr2line traslada 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

El programa ar crea, modifica y extrae desde archivos. Un archivo es un fichero que almacena una colección de otros ficheros en una estructura que hace posible obtener el original de cada fichero individual (llamados miembros del archivo).

as

as está pensado, principalmente, para ensamblar la salida del compilador GNU, gcc, para ser usada por el enlazador ld.

gprof

gprof muestra el grafo de llamadas de los datos perfilados.

ld

ld combina un número de objetos y ficheros de archivo, reubica sus datos y establece las referencias a los símbolos. Frecuentemente, el último paso de la compilación de un nuevo programa es hacer una llamada a ld.

nm

nm lista los símbolos de los ficheros objeto.

objcopy

La utilidad objcopy copia el contenido de un fichero objeto en otro. objcopy usa la librería BFD de GNU para leer y escribir los ficheros objeto. Puede escribir el fichero objeto destino en un formato diferente al del fichero objeto fuente.

objdump

objdump muestra información sobre uno o más ficheros objeto. Mediante opciones se puede indicar la información a mostrar. Esta información es útil fundamentalmente para los programadores que trabajan en herramientas de compilación (al contrario de los programadores que sólo quieren que sus programas compilen y funcionen).

ranlib

ranlib genera un índice de los contenidos de un archivo, y lo coloca en el archivo. El índice lista cada símbolo definido por un miembro de un archivo que es un fichero objeto reubicable.

readelf

readelf muestra información sobre binarios de tipo elf.

size

size lista los tamaños de las secciones −−y el tamaño total−− para cada uno de los ficheros objeto en su lista de argumentos. Por defecto, se genera una línea de salida por cada fichero objeto o cada módulo de un archivo.

strings

Para cada fichero dado, strings muestra las cadenas de caracteres imprimibles de al menos 4 caracteres (o el número especificado en las opciones del programa) seguidas por un carácter no imprimible. Por defecto, sólo muestra las cadenas procedentes de las secciones de inicialización y carga de los ficheros objeto; para otros tipos de ficheros muestra todas las cadenas de los mismos.

strings es útil, principalmente, para determinar el contenido de ficheros que no son de texto.

strip

strip elimina todos los símbolos o sólo los especificados de los ficheros objeto. La lista de ficheros objeto puede incluir archivos. Se debe indicar, por lo menos, un fichero objeto. strip modifica los ficheros mencionados en sus argumentos, en vez de escribir copias modificadas con otro nombre.

Descripción de las librerías

libbfd

libbfd es la librería de descriptores de ficheros binarios (BFD).

libopcodes

libopcodes es una librería nativa para manejar mnemónicos y se usa durante la construcción de utilidades como objdump. Los mnemónicos son, en realidad, las versiones en texto legible de las instrucciones del procesador.

Dependencias de instalación de Binutils

Última versión comprobada: 2.11.2.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh

Binutils: ar, as, ld, nm, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, rmdir, touch
Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, true, uname
Texinfo: install−info, makeinfo
Textutils: cat, sort, tr, uniq

Bison

Localización oficial para descarga

Bison (1.875):
ftp://ftp.gnu.org/gnu/bison/

Contenido de Bison

Última versión comprobada: 1.35.

Bison es un generador de analizadores sintácticos, un sustituto de yacc. Bison genera un programa que analiza la estructura de un fichero de texto.

Bison instala lo siguiente:

Programas

bison y yacc

Descripciones

Última versión comprobada: 1.35.

Descripción de los programas

bison

bison es un generador de analizadores sintácticos, un sustituto de yacc ("Yet Another Compiler Compiler", Otro Compilador de Compiladores). Entonces, ¿qué es bison? Es un programa que genera un programa que analiza la estructura de un fichero de texto. En lugar de escribir el programa, el usuario especifica qué cosas deben ser conectadas y con esas reglas se genera un programa que analiza el fichero de texto. Hay muchos

ejemplos en los que se necesita esta estructura y uno de ellos es la calculadora.

Tomando la cadena :

```
1 + 2 * 3
```

Una persona puede fácilmente saber que el resultado es 7. ¿Por qué? Porque al ver la estructura nuestro cerebro sabe como interpretar la cadena. La computadora no sabe eso y bison es una herramienta que le ayuda a interpretarla presentando la cadena de la siguiente forma al compilador:

```
  +
 / \
 *  1
/ \
2  3
```

Comenzando por la base del árbol y subiendo por los números 2 y 3, que están unidos por el símbolo de la multiplicación, la computadora multiplica 2 y 3. Almacena el resultado de la multiplicación y, lo siguiente que ve, es el resultado de 2*3 y el número 1 unido con el símbolo de la suma. Añadiendo 1 al resultado previo se obtiene 7. El cálculo de formulas más complejas puede hacerse pasándolas a este formato de árbol. El ordenador comienza justo por la base y sigue trabajando hacia arriba hasta alcanzar el resultado correcto. Por supuesto, bison no se usa sólo en calculadoras.

yacc

Este guión de bash invoca a bison usando la opción −y. Esto es por compatibilidad con programas que usan yacc en lugar de bison.

Dependencias de instalación de Bison

Última versión comprobada: 1.31.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, head, tr, uniq

Bzip2

Localización oficial para descarga

Bzip2 (1.0.2):
http://sources.redhat.com/bzip2/

Contenido de Bzip2

Última versión comprobada: 1.0.2

Bzip2 es un compresor de ficheros por ordenación de bloques que, generalmente, consige una mejor compresión que el tradicional **`gzip`**.

Bzip2 instala lo siguiente:

Programas

bunzip2 (enlace a bzip2), bzcat (enlace a bzip2), bzcmp, bzdiff, bzegrep, bzfgrep, bzgrep, bzip2, bzip2recover, bzless y bzmore

Librerías

libbz2.a, libbz2.so (enlace a libbz2.so.1.0), libbz2.so.1.0 (enlace a libbz2.so.1.0.2) y libbz2.so.1.0.2

Descripciones

Última versión comprobada: 1.0.2

Descripción de los programas

bunzip2

bunzip2 descomprime ficheros que han sido comprimidos con bzip2.

bzcat

bzcat (o bzip2 −dc) descomprime todos los ficheros especificados hacia la salida estándar.

bzcmp, bzdiff

bzcmp y bzdiff se usan para invocar el programa cmp o diff en ficheros comprimidos con bzip2.

bzegrep, bzfgrep, bzgrep

bzegrep, bzfgrep, y bzgrep invocan, respectivamente, a egrep, fgrep, o grep en ficheros comprimidos con bzip2.

bzip2

bzip2 comprime ficheros usando el algoritmo de compresión de texto por ordenación de bloques Burrows−Wheeler y la codificación Huffman. La compresión es, en general, considerablemente superior a la obtenida por otros compresores más convencionales basados en el LZ77/LZ78 y se acerca al rendimiento de la familia de compresores estadísticos PPM.

bzip2recover

bzip2recover recupera datos de ficheros bzip2 dañados.

bzless

bzless es un filtro que permite examinar ficheros comprimidos o de texto plano, pantalla a pantalla en un terminal emulado, como less.

bzmore

bzmore es un filtro que permite examinar ficheros comprimidos o de texto plano, pantalla a pantalla en un terminal emulado, como more.

Descripción de las librerías

libbz2

libbz2 es la librería que implementa la compresión sin pérdidas por ordenación de bloques, usando el algoritmo de Burrows−Wheeler.

Dependencias de instalación de Bzip2

Última versión comprobada: 1.0.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Fileutils: cp, ln, rm
Gcc: cc1, collect2, cpp0, gcc
Make: make

Diffutils

Localización oficial para descarga

Diffutils (2.8.1):
ftp://ftp.gnu.org/gnu/diffutils/

Contenido de Diffutils

Última versión comprobada: 2.8.1.

Los programas de este paquete te muestran las diferencias entre dos ficheros o directorios. Es muy común usarlos para crear parches de software.

Diffutils instala lo siguientes:

Programas

cmp, diff, diff3 y sdiff

Descripciones

Última versión comprobada: 2.8.1.

Descripción de los programas

cmp y diff

Tanto cmp como diff comparan dos ficheros y muestran sus diferencias. Ambos programas tienen argumentos para comparar ficheros en diferentes situaciones.

diff3

La diferencia entre diff y diff3 es que diff compara 2 ficheros mientras diff3 compara 3.

sdiff

sdiff mezcla dos ficheros y muestra los resultados interactivamente.

Dependencias de instalación de Diffutils

Última versión comprobada: 2.7.

Bash: sh
Binutils: ld, as
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed

Sh−utils: date, hostname
Textutils: cat, tr

E2fsprogs

Localización oficial para descarga

E2fsprogs (1.32):
ftp://download.sourceforge.net/pub/sourceforge/e2fsprogs/
http://download.sourceforge.net/e2fsprogs/

Contenido de E2fsprogs

Última versión comprobada: 1.27.

E2fsprogs proporciona las utilidades para los sistemas de ficheros ext2. También soporta los sistemas de ficheros ext3 con registro de transaciones.

E2fsprogs instala lo siguiente:

Programas

badblocks, chattr, compile_et, debugfs, dumpe2fs, e2fsck, e2image, e2label, fsck, fsck.ext2, fsck.ext3, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mklost+found, resize2fs, tune2fs y uuidgen

Librerías

libcom_err.[a,so], libe2p.[a,so], libext2fs.[a,so], libss.[a,so], libuuid.[a,so]

Descripciones

Última versión comprobada: 1.27.

Descripción de los programas

badblocks

badblocks se usa para buscar bloques dañados en un dispositivo (normalmente una partición de disco).

chattr

chattr cambia los atributos de un fichero en un sistema de ficheros ext2 de Linux.

compile_et

compile_et es usado para convertir una tabla con códigos de error y sus mensajes asociados en un fichero fuente C apropiado para usar con la librería com_err.

debugfs

El programa debugfs es un depurador de sistemas de ficheros. Puede usarse para examinar y cambiar el estado de un sistema de ficheros ext2.

dumpe2fs

dumpe2fs muestra la información del superbloque y de los grupos de bloques del sistema de ficheros presente en un determinado dispositivo.

e2fsck y fsck.ext2

e2fsck y fsck.ext2 se usan para chequear, y opcionalmente reparar, sistemas de ficheros ext2.

e2image

e2image se usa para salvar información crítica de un sistema de ficheros ext2 en un fichero.

e2label

e2label muestra o cambia la etiqueta de un sistema de ficheros ext2 situado en el dispositivo especificado.

fsck

fsck se usa para chequear, y opcionalmente reparar, un sistema de ficheros Linux.

fsck.ext3

fsck.ext3 se usa para chequear, y opcionalmente reparar, un sistema de ficheros ext3.

lsattr

lsattr muestra los atributos de un fichero en un sistema de ficheros ext2.

mk_cmds

La utilidad mk_cmds toma como entrada un fichero de tabla de comandos y genera como salida un fichero fuente C preparado para usarlo con la librería del subsistema, libss.

mke2fs and mkfs.ext2

mke2fs se usa para crear sistemas de ficheros ext2 en un dispositivo (normalmente una partición de disco). mkfs.ext2 hace lo mismo que mke2fs.

mkfs.ext3

mkfs.ext3 se usa para crear un sistema de ficheros ext3.

mklost+found

mklost+found se usa para crear un directorio lost+found en el directorio de trabajo actual de un sistema de ficheros ext2. mklost+found reserva una serie de bloques de disco en el directorio para que sean usados por e2fsck.

resize2fs

resize2fs se usa para redimensionar sistemas de ficheros ext2.

tune2fs

tune2fs ajusta los parámetros de un sistema de ficheros ext2.

uuidgen

El programa uuidgen crea un nuevo identificador universal único (UUID) usando la librería libuuid. El 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.

Descripción de las librerías

libcom_err

Rutinas para mostrar errores comunes.

libe2p

libe2p es usada por dumpe2fs, chattr, y lsatt.

libext2fs

La librería libext2fs esá diseñada para permitir a los programas de nivel de usuario manipular un sistema de ficheros ext2.

libss

libss es usada por debugfs.

libuuid

La librería libuuid se usa para generar identificadores únicos para objetos que pueden estar accesibles más allá del sistema local.

Dependencias de instalación de E2fsprogs

Última versión comprobada: 1.25.

Bash: sh
Binutils: ar, as, ld, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mkdir, mv, rm, sync
Gcc: cc, cc1, collect2, cpp0
Glibc: ldconfig
Grep: egrep, grep

Gzip: gzip
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, echo, expr, hostname, uname
Texinfo: makeinfo
Textutils: cat, tr

Ed

Localización oficial para descarga

Ed (0.2):
ftp://ftp.gnu.org/gnu/ed/

Parche para Ed (0.2):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Ed

Última versión comprobada 0.2.

GNU ed es un editor de líneas de 8 bits limpio y que cumple con POSIX.

Ed instala lo siguiente:

Programas

ed y red (enlace a ed)

Descripciones

Última versión comprobada: 0.2.

Descripción de los programas

ed

ed es un editor de líneas de texto. Se usa para crear, mostrar, modificar o cualquier otra manipulación de ficheros de texto.

red

red es un ed restringido: sólo puede editar ficheros del directorio actual y no puede ejecutar comandos del intérprete de comandos.

Dependencias de instalación de Ed

Última versión comprobada: 0.2.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: hostname
Textutils: cat, tr

File

Localización oficial para descarga

File (3.39):
ftp://ftp.gw.com/mirrors/pub/unix/file/

Contenido de File

Última versión comprobada: 3.39.

File es una utilidad usada para determinar el tipo de los ficheros.

File instala lo siguiente:

Programas

file

Descripciones

Última versión comprobada: 3.39.

Descripción del programa

file

file comprueba cada fichero especificado para clasificarlo. Se hacen tres tipos de pruebas, en este orden: pruebas de sistemas de ficheros, pruebas de números mágicos y pruebas de lenguajes. La primera prueba que

tenga éxito hace que se muestre el tipo de fichero.

Dependencias de instalación de File

Última versión comprobada: 3.37.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: echo, expr, hostname, sleep
Texinfo: makeinfo
Textutils: cat, tr

Fileutils

Localización oficial para descarga

Fileutils (4.1):
ftp://ftp.gnu.org/gnu/fileutils/

Parche para Fileutils (4.1):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Fileutils

Última versión comprobada: 4.1.

Fileutils es un paquete que contiene los programas básicos para la manipulación de ficheros. Incluye programas para listar y crear directorios, actualizar las marcas de fechas, cambiar los permisos y más.

Fileutils instala lo siguiente:

Programas

chgrp, chmod, chown, cp, dd, df, dir, dircolors, du, install, ln, ls, mkdir, mkfifo, mknod, mv, rm, rmdir, shred, sync, touch y vdir

Descripciones

Última versión comprobada: 4.1.

Descripción de los programas

chgrp

chgrp cambia el grupo de cada fichero al grupo especificado, que puede ser tanto el nombre de un grupo como su identificador numérico.

chmod

chmod cambia los permisos de un fichero de acuerdo con el modo, que puede ser tanto una representación simbólica de los cambios a hacer o un número octal que representa el patrón de bits de los nuevos permisos.

chown

chown cambia el usuario y/o el grupo al que pertenece un fichero.

cp

cp copia ficheros de un lugar a otro.

dd

dd copia un fichero (por defecto, de la entrada estándar a la salida estándar) con un tamaño de bloque definido por el usuario, mientras, opcionalmente, realiza conversiones en él.

df

df muestra la cantidad de espacio disponible en los sistemas de ficheros a los que pertenece cada fichero que se le pasa como argumento. Si no se indica ningún fichero, se muestra el espacio disponible en todos los sistemas de ficheros montados actualmente

dir, ls y vdir

dir y vdir son versiones de ls con formatos de salida diferentes. Estos programas listan cada fichero o directorio especificado. El contenido de los directorios se lista alfabéticamente. Para ls, los ficheros se listan, por defecto, en columnas ordenados verticalmente si la salida estándar es un terminal; en otro caso se listan uno por línea. Para dir, los ficheros se listan, por defecto, en columnas ordenados verticalmente. Para vdir, los ficheros se listan, por defecto, en formato largo.

dircolors

dircolors imprime comandos para modificar la variable de entorno LS_COLOR, que se usa para cambiar el esquema de color por defecto de ls y de herramientas relacionadas.

du

du muestra la cantidad de espacio en disco usado por cada fichero o directorio listado en la línea de comandos, y por cada uno de sus subdirectorios.

install

install copia ficheros y establece sus permisos y, si es posible, su propietario y grupo.

ln

ln crea enlaces duros o blandos (simbólicos) entre ficheros.

mkdir

mkdir crea directorios con el nombre indicado.

mkfifo

mkfifo crea una tubería (FIFO) con un nombre dado.

mknod

mknod crea una tubería (FIFO), un fichero especial de caracteres o un fichero especial de bloques con el nombre indicado.

mv

mv mueve ficheros de un directorio a otro o renombra ficheros, dependiendo de los argumentos que se le pasen.

rm

rm elimina ficheros o directorios.

rmdir

rmdir elimina directorios, si están vacíos.

shred

shred borra un fichero de forma segura, sobreescribiéndolo para que su contenido no pueda ser recuperado.

sync

sync guarda los bloques modificados en disco y actualiza el superbloque.

touch

touch cambia las fechas de modificación o acceso del fichero especificado, poniéndole la fecha actual. Si el fichero no existe crea uno vacío.

Dependencias de instalación de Fileutils

Última versión comprobada: 4.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
Make: make
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, tr

Findutils

Localización oficial para descarga

Findutils (4.1):
ftp://ftp.gnu.org/gnu/findutils/

Parche para Findutils (4.1):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Parche de Violación de Segmento para Findutils (4.1):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Findutils

Última versión comprobada: 4.1.

El paquete Findutils contiene pogramas para encontrar ficheros, tanto al vuelo (haciendo una búsqueda recursiva en vivo a través de los directorios y mostrando sólo los ficheros que cumplan las especificaciones) o mediante una búsqueda a través de una base de datos.

Findutils instala lo siguiente:

Programas

bigram, code, find, frcode, locate, updatedb y xargs

Descripciones

Última versión comprobada: 4.1.

Descripción de los programas

bigram

bigram se usa junto con code para generar las bases de datos de locate en el formato antiguo. Para saber más de estos tres programas, lee la página de manual locatedb.5.

code

code es el antecesor de frcode. Se usaba en las bases de datos de formato antiguo.

find

El programa find busca los ficheros de una jerarquía de directorios que cumplan un cierto criterio. Si no se especifica un criterio de búsqueda, lista todos los ficheros del directorio actual y de los subdirectorios.

frcode

frcode es llamado por updatedb para comprimir la lista de ficheros usando "front−compression", que reduce el tamaño de la base de datos en un factor de 4 o 5.

locate

locate busca en una base de datos que contiene todos los ficheros y directorios de un sistema de ficheros. Este programa lista los ficheros y directorios de la base de datos que cumplan cierto criterio. Si un usuario busca un fichero, este programa buscará en la base de datos y le dirá dónde están ubicados exactamente esos ficheros. Esto sólo es válido si la base de datos de locate se encuentra actualizada. En otro caso mostrará información anticuada.

updatedb

El programa 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 directorios y ficheros que encuentre en la base de datos que usa locate para recuperar dicha información. Es una buena costumbre actualizar la base de datos una vez al día para obtener información correcta cuando se necesite.

xargs

El comando xargs aplica un comando a una lista de ficheros. Si se necesita aplicar el mismo comando sobre múltiples ficheros, puede crearse una lista que contenga todos estos ficheros (uno por línea) y xargs puede aplicar dicho comando en esos ficheros.

Dependencias de instalación de Findutils

Última versión comprobada: 4.1.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Grep: egrep, grep
Gcc: cc1, collect2, cpp0, gcc
Make: make
Patch: patch
Sed: sed
Sh−utils: basename, date, echo, hostname
Textutils: cat, tr

Flex

Localización oficial para descarga

Flex (2.5.4a):
ftp://ftp.gnu.org/non−gnu/flex/

Contenido de Flex

Última versión comprobada: 2.5.4a.

El pquete Flex se utiliza para generar programas que reconocen patrones de texto.

Flex instala lo siguiente:

Programas

flex, flex++ (enlace a flex) y lex

Librerías

libfl.a

Descripciones

Última versión comprobada: 2.5.4a.

Descripción de los programas

flex

flex es una herramienta para generar programas capaces de reconocer patrones de texto. El reconocimiento de patrones es muy útil en muchas aplicaciones. El usuario establece las reglas de búsqueda y flex generará el programa que buscará esos patrones. La razón por la que la gente usa flex es porque es mucho más fácil establecer las reglas de búsqueda que escribir un programa real que busque el texto.

flex++

flex++ invoca una versión de flex usada exclusivamente para generar analizadores en C++.

lex

Creamos un guión de bash llamado lex que invoca a flex con la opción −l. Esto es por compatibilidad con programas que usan lex en lugar de flex

Descripción de las librerías

libfl

lbfl es la librería flex.

Dependencias de instalación de Flex

Última versión comprobada: 2.5.4a.

Bash: sh
Binutils: ar, as, ld, ranlib
Bison: bison
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mv, rm, touch
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: echo, hostname
Textutils: cat, tr

Gawk

Localización oficial para descarga

Gawk (3.1.1):
ftp://ftp.gnu.org/pub/gnu/gawk/

Parche para Gawk (3.1.1–3):
http://downloads.linuxfromscratch.org/

Contenido de Gawk

Última versión comprobada: 3.1.1.

Gawk es uns implementación de awk utilizada para manipular ficheros de texto.

Gawk instala lo siguiente:

Programas

awk, gawk, gawk–3.1.1, grcat, igawk, pgawk, pgawk–3.1.1, pwcat

Descripciones

Última versión comprobada: 3.1.1.

Descripción de los programas

awk

awk es un enlace simbólico a gawk.

gawk, gawk–3.1.1

gawk es la implementación GNU de awk, un explorador de patrones y procesador de lenguajes.

grcat

grcat concatena la base de datos de grupos, /etc/group.

igawk

igawk es un guión del interprete de comandos que otorga a gawk la capacidad de incluir ficheros.

pgawk, pgawk–3.1.1

pgawk es la versión de gawk con soporte de perfiles.

pwcat

pwcat concatena la base de datos de contraseñas, /etc/passwd.

Dependencias de instalación de Gawk

Última versión comprobada: 3.1.0.

(Todavía no se han comprobado las dependencias.)

GCC

Localización oficial para descarga

GCC (3.2.1):
ftp://ftp.gnu.org/pub/gnu/gcc/

Contenido de GCC

Última versión comprobada: 3.1.

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

GCC instala lo siguiente:

Programas

c++, c++filt, cc (enlace a gcc), cc1, cc1plus, collect2, cpp, cpp0, g++, gcc, gccbug, gcov y tradcpp0

Librerías

libgcc.a, libgcc_eh.a, libgcc_s.so, libiberty.a, libstdc++.[a,so], libsupc++.a

Descripciones

Última versión comprobada: 3.1.

Descripción de los programas

cc, cc1, cc1plus, gcc

Estos programas forman el compilador de C. Un compilador convierte el código fuente en formato de texto a un formato que un ordenador pueda entender. Después de que un fichero de código fuente es compilado en un fichero objeto, un enlazador creará un fichero ejecutable a partir de uno o más de estos ficheros objeto generados por el compilador.

c++, cc1plus, g++

Estos programas forman el compilador de C++, el equivalente de cc, gcc, etc.

c++filt

El lenguaje C++ proporciona sobrecarga de funciones, lo que significa que es posible escribir varias funciones con el mismo nombre (suponiendo que cada una tome parámetros de diferente tipo). Todos los nombres de funciones C++ son codificadas dentro de una etiqueta de bajo nivel del ensamblador (este proceso es conocido como "mangling"). El programa c++flit hace lo contrario: decodifica (demangling) nombres de bajo nivel en nombres de nivel de usuario para que el enlazador pueda evitar conflictos en estas funciones sobrecargadas.

collect2

collect2 asiste en la compilación de constructores.

cpp, cpp0

cpp preprocesa un fichero fuente, por ejemplo la inclusión del contenido de los ficheros de cabecera en el fichero fuente. Simplemente, añade una línea del tipo #include <fichero> a tu fichero fuente y el preprocesador insertará el contenido del fichero incluido dentro del fichero fuente.

gccbug

gccbug es un guión del interprete de comandos que se usa para simplificar la creación de notificaciones de errores.

gcov

gcov analiza programas para ayudar a crear códigos más eficentes y más rápidos mediante optimizaciones.

tradcpp0

No hay descripción disponible.

Descripción de las librerías

libgcc, libgcc_eh, libgcc_s

Ficheros de soporte en tiempo de ejecución para gcc.

libiberty

libiberty es una colección de subrutinas usadas por muchos programas GNU, incluidos getopt, obstack, strerror, strtol y strtoul.

libstdc++

libstdc++ es la librería C++. Es utilizada por programas escritos en C++ y contiene funciones que son usadas frecuentemente por esos programas. De esta forma el programador no necesita escribir ciertas funciones (como la escritura de una cadena de texto en pantalla) desde el principio cada vez que crea un programa.

libsupc++

libsupc++ proporciona soporte para el lenguaje de programación c++. Entre otras cosas, libsupc++ contiene rutinas para el manejo de excepciones.

Dependencias de instalación de GCC

Última versión comprobada: 2.95.3.

Bash: sh
Binutils: ar, as, ld, nm, ranlib
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch
Find: find
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: basename, dirname, echo, expr, hostname, sleep, true, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tail, tr

Gettext

Localización oficial para descarga

Gettext (0.11.5):
ftp://ftp.gnu.org/gnu/gettext/

Contenido de Gettext

Última versión comprobada: 0.11.2.

El paquete Gettext se utiliza para la internacionalización y localización. Los programas pueden compilarse con Soporte de Lenguaje Nativo (NLS), lo que les permite mostrar mensajes en el idioma nativo del usuario.

Gettext instala lo siguiente:

Programas

config.charset, config.rpath, gettext, gettextize, hostname, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, project−id, team−address, trigger, urlget, user−email y xgettext

Librerías

libgettextlib[a,so], libgettextsrc[a,so]

Descripciones

Última versión comprobada: 0.11.2.

Descripción de los programas

config.charset

El guión config.charset saca una tabla dependiente del sistema de los alias de codificación de los caracteres.

config.rpath

El guión config.rpath saca un grupo de variables dependientes del sistema, describiendo cómo fijar la ruta de busqueda en tiempo de ejecución de las librerías compartidas en un ejecutable.

gettext

El paquete gettext se usa en la internacionalización (también conocida como i18n) y en la localización (conocida como l10n). Los programas pueden ser compilados con soporte para lenguaje nativo (NLS) que activa el que se muestren los mensajes de salida en el idioma del usuario en vez de en el idioma por defecto, el inglés.

gettextize

El programa gettextize copia todos los ficheros estándar gettext en un directorio. Se usa para hacer paquetes con traducción gettext.

hostname

El programa hostname muestra el nombre de un sistema en la red en varios formatos.

msgattrib

El programa msgattrib filtra los mensajes de un catálogo de traducción de acuerdo con sus atributos, y manipula dichos atributos.

msgcat

El programa msgcat encuentra los mensajes comunes en varias traducciones directas.

msgcmp

El programa msgcmp compara dos ficheros de traducción directa.

msgcomm

El programa msgcomm busca los mensajes que aparecen en varios ficheros .po. Se usa para comparar cómo deben traducirse las cosas.

msgconv

El programa msgconv convierte un catálogo de traducción a una codificación de caracteres diferente.

msgen

El programa msgen crea un catálogo de traducción en inglés.

msgexec

El programa msgexec aplica un comando a todas las traducciones de un catálogo de traducción.

msgfilter

El programa msgfilter aplica un filtro a todas las traducciones de un catálogo de traducción.

msgfmt

El programa msgfmt compila traducciones directas en código máquina. Se usa para crear el fichero de traducción final de un programa/paquete.

msggrep

El programa msggrep extrae todos los mensajes de un catálogo de traducción que cumplan cierto criterio o pertenezcan a alguno de los ficheros fuente indicados.

msginit

El programa msginit crea un nuevo fichero PO, inicializando la información con valores procedentes del entorno del usuario.

msgmerge

El programa msgmerge combina dos traducciones directas en un fichero. Se usa para actualizar la traducción directa con el extracto de las fuentes.

msgunfmt

El programa msgunfmt descompila ficheros de traducciones en traducciones directas de texto. Sólo puede ser usado si la versión compilada está disponible.

msguniq

El programa msguniq unifica las traducciones duplicadas en un catálogo de traducción.

ngettext

El programa ngettext muestra traducciones en lenguaje nativo de un mensaje de texto cuya forma gramatical depende de un número.

project−id

El guión project−id imprime una identificación de la versión de un paquete o del paquete.

team−address

El guión team−address imprime las direcciones del equipo de desarrolladores en la salida estándar, y muestra instrucciones adicionales .

trigger

El guión trigger comprueba si el paquete actual es un paquete de GNOME o de KDE.

urlget

El programa urlget captura el contenido de una URL.

user−email

El guión user−email imprime la dirección de correo electrónico de un usuario, con la confirmación del usuario.

xgettext

El programa xgettext extrae las líneas de mensajes de los ficheros C del programador. Se usa para hacer la primera plantilla de traducción.

Descripción de las librerías

libgettextlib

No hay descripción disponible.

libgettextsrc

No hay descripción disponible.

Dependencias de instalación de Gettext

Última versión comprobada: 0.10.40.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh

Binutils: ar, as, ld, nm, ranlib, strip
Bison: bison
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm, rmdir
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, sort, tr, uniq

Glibc

Localización oficial para descarga

Glibc (2.3.1):
ftp://ftp.gnu.org/gnu/glibc/

Glibc−linuxthreads (2.3.1):
ftp://ftp.gnu.org/gnu/glibc/

Parche Root/Perl para Glibc (2.3.1):
http://downloads.linuxfromscratch.org/

Parche Libnss para Glibc (2.3.1):
http://downloads.linuxfromscratch.org/

Contenido de Glibc

Última versión comprobada: 2.2.5.

Glibc es la librería C que proporciona las llamadas al sistema y las funciones básicas, tales como open, malloc, printf, etc. La librería C es utilizada por todos los programas enlazados dinámicamente.

Glibc instala lo siguiente:

Programas

catchsegv, gencat, getconf, getent, glibcbug, iconv, iconvconfig, ldconfig, ldd, lddlibc4, locale, localedef, mtrace, nscd, nscd_nischeck, pcprofiledump, pt_chown, rpcgen, rpcinfo, sln, sprof, tzselect, xtrace, zdump y zic

Librerías

ld.so, libBrokenLocale.[a,so], libSegFault.so, libanl.[a,so], libbsd−compat.a, libc.[a,so], libc_nonshared.a, libcrypt.[a,so], libdl.[a,so], libg.a, libieee.a, libm.[a,so], libmcheck.a, libmemusage.so, libnsl.a, 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

Última versión comprobada: 2.2.5.

Descripción de los programas

catchsegv

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

gencat

gencat genera catálogos de mensajes.

getconf

getconf muestra los valores de configuración del sistema para variables especificas del sistema de ficheros.

getent

getent obtiene entradas de una base de datos administrativa.

glibcbug

glibcbug crea un informe de fallos sobre glibc y lo envía a la dirección de correo electrónico de errores.

iconv

iconv realiza conversiones de los juegos de caracteres.

iconvconfig

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

ldconfig

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

ldd

ldd muestra las librerías compartidas requeridas por cada programa o librería especificada en la línea de comandos.

lddlibc4

lddlibc4 asiste a ld con los ficheros objeto.

locale

locale es un programa Perl que le dice al compilador si debe activar (o desactivar) el uso de las locales POSIX para operaciones integradas.

localedef

localedef compila las especificaciones para locale.

mtrace

No hay deecripción disponible.

nscd

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

nscd_nischeck

nscd_nischeck comprueba si es necesario o no un modo seguro para búsquedas NIS+.

pcprofiledump

pcprofiledump vuelca la información generada por un perfil de PC.

pt_chown

pt_chown establece el propietario, grupo y permisos de acceso del pseudo−terminal esclavo correspondiente al pseudo−terminal maestro apuntado por el descriptor de ficheros "3". Este es el programa de ayuda para la función `grantpt'. No está pensado para ejecutarse directamente desde la línea de comandos.

rpcgen

rpcgen genera código C para implementar el protocolo RPC.

rpcinfo

rpcinfo hace una llamada RPC en un servidor RPC.

sln

sln enlaza simbólicamente un destino a una fuente. Está enlazado estáticamente, no necesitando enlazado dinámico. Por tanto, sln es útil para crear enlaces simbólicos a librerías dinámicas si, por alguna razón, el enlazador dinámico del sistema no funciona.

sprof

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

tzselect

tzselect pregunta al usuario información sobre la localización actual y muestra la descripción de la zona horaria resultante en la salida estándar.

xtrace

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

zdump

zdump es el visualizador de información de huso horario.

zic

zic es el compilador de la zona horaria.

Descripción de las librerías

ld.so

ld.so es el programa de ayuda para las librerías compartidas ejecutables.

libBrokenLocale

Usadas por software como Mozilla para resolver locales rotas.

libSegFault

libSegFault es un manejador de señales de violación de segmento. Intenta capturar estas señales.

libanl

libanl es una librería de búsqueda de nombres asíncrona.

libbsd−compat

libbsd−compat proporciona la portabilidad necesaria para ejecutar ciertos programas en Linux.

libc, libc_nonshared

Estos ficheros constituyen la librería C principal, que es una colección de funciones usadas frecuentemente en programas. De esta forma un programador no necesita crear sus propias funciones para cada tarea individual. Las cosas más comunes, como mostrar una cadena en pantalla, están presentes y a disposición del programador.

La librería C (en realidad, casi todas las librerías) viene en dos formas: dinámica y estática. En resumen, cuando un programa usa una librería C estática, se copia el código de la librería C dentro del ejecutable. Cuando un programa usa una librería dinámica, el ejecutable no contiene el código de la librería pero, en su lugar, tiene una rutina que carga las funciones desde esa librería en el momento en el que se ejecuta. De esta forma disminuye de forma significativa el tamaño del programa. La documentación que acompaña a la librería C describe esto con más detalle, pues es demasiado complicado explicarlo aquí en dos o tres líneas.

libcrypt

libcrypt es la librería criptográfica.

libdl

libdl es la librería de interfaz del enlazado dinámico.

libg

libg es una librería en tiempo de ejecución de g++.

libieee

libieee es la librería de punto flotante IEEE.

libm

libm es la librería matemática.

libmcheck

libmcheck contiene código ejecutado en el arranque.

libmemusage

libmemusage es usada por memusage para ayudar a recoger información sobre el uso de memoria de un programa.

libnsl

libnsl es la librería de servicios de red.

libnss_compat, libnss_dns, libnss_files, libnss_hesiod, libnss_nis, libnss_nisplus

La idea básica es poner en módulos separados la implementación de los diferentes servicios ofrecidos para acceder a las bases de datos. Esto tiene algunas ventajas:

- los colaboradores pueden añadir nuevos servicios sin tener que añadirlos a la librería C de GNU,
- los módulos pueden actualizarse separadamente,
- la imagen de la librería C es más pequeña.

libpcprofile

Código usado por el núcleo para rastrear el tiempo de CPU gastado en funciones, líneas de código fuente e instrucciones.

libpthread

La librería de hilos POSIX.

libresolv

Esta librería proporciona funciones para la creación, envío e interpretación de paquetes de datos a servidores de nombres de dominio de Internet.

librpcsvc

Esta librería proporciona funciones para una miscelánea de servicios RPC.

librt

Esta librería proporciona funciones para muchas de las interfaces especificadas por el POSIX.1b Realtime Extension (Extensiones en Tiempo Real POSIX.1b).

libthread_db

Las funciones de esta librería son útiles para construir depuradores para programas multihilo.

libutil

Contienen código para funciones "estándar" usadas en diferentes utilidades Unix.

Dependencias de instalación de Glibc

Última versión comprobada: 2.2.5.

Bash: sh
Binutils: ar, as, ld, ranlib, readelf
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mknod, mv, mkdir, rm, touch
Gcc: cc, cc1, collect2, cpp, gcc
Grep: egrep, grep
Gzip: gzip
Make: make
Gawk: gawk
Sed: sed
Sh−utils: date, expr, hostname, pwd, uname
Texinfo: install−info, makeinfo
Textutils: cat, cut, sort, tr

Grep

Localización oficial para descarga

Grep (2.5):
ftp://ftp.gnu.org/gnu/grep/

Contenido de Grep

Última versión comprobada: 2.5.

Grep es un programa usado para imprimir las líneas de un fichero que cumplan un patrón especificado.

Grep instala lo siguiente:

Programas

egrep (enlace a grep), fgrep (enlace a grep) y grep

Descripciones

Última versión comprobada: 2.5.

Descripción de los programas

egrep

egrep muestra las líneas de un fichero que coincidan con una determinada expresión regular extendida.

fgrep

fgrep muestra las líneas de un fichero que coincidan con una lista de cadenas fijas, separadas por saltos de línea, cualquiera de las cuales puede ser coincidente.

grep

grep muestra las líneas de un fichero que coincidan con una expresión regular.

Dependencias de instalación de Grep

Última versión comprobada: 2.4.2.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf

Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Groff

Localización oficial para descarga

Groff (1.18.1):
ftp://ftp.gnu.org/gnu/groff/

Contenido de Groff

Úlrima versión comprobada: 1.17.2.

El paquete Groff incluye varios programas de procesamiento de texto para formatear el texto. Grof traduce el texto estándar y los comandos especiales a salida formateada, como la que puedes ver en las páginas de manual.

Groff instala lo siguiente:

Programas

addftinfo, afmtodit, eqn, geqn (enlace a eqn), grn, grodvi, groff, grog, grolbp, grolj4, grops, grotty, gtbl (enlace a tbl), hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pfbtops, pic, post−grohtml, pre−grohtml, refer, soelim, tbl, tfmtodit, troff y zsoelim (enlace a soelim)

Descripciones

Última versión comprobada: 1.17.2.

Descripción de los programas

addftinfo

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

afmtodit crea un fichero de fuentes para usarlo con groff y grops.

eqn

eqn compila las descripciones de las formulas embebidas en los ficheros de entrada de troff a comandos que pueda entender troff.

geqn

geqn es la implementación GNU de eqn.

grn

grn es un preprocesador groff para ficheros gremlin.

grodvi

grodvi es un controlador para groff que genera formatos dvi de TeX.

groff

groff es una interfaz para el sistema de formateado de documentos groff. Normalmente lanza el programa troff y un post−procesador apropiado para el dispositivo seleccionado.

grog

grog lee ficheros y averigua cual de las opciones −e, −man, −me, −mm, −ms, −p, −s, y −t de groff se necesitan para mostrar los ficheros, y muestra el comando de groff incluyendo esas opciones en la salida estándar.

grolbp

grolbp es un controlador de groff para las impresoras Canon CAPSL (series LBP−4 y LBP−8 de impresoras láser)

grolj4

grolj4 es un controlador para groff que produce salidas en el formato PCL5 adecuado para impresoras HP Laserjet 4.

grops

grops transforma la salida de GNU troff en Postscript.

grotty

grotty transforma la salida de GNU troff en un formato adecuado para dispositivos tipo máquina de escribir.

gtbl

gtbl es la implementación GNU de tbl.

hpftodit

hpftodit crea un fichero de fuentes para usar con groff −Tlj4 a partir de ficheros de marcas de fuentes métricas de HP.

indxbib

indxbib hace un índice inverso para la base de datos bibliográfica, un fichero específico para usarlo con refer, lookbib, y lkbib.

lkbib

lkbib busca, en las bases de datos bibliográficas, referencias que contengan las claves especificadas y muestra cualquier referencia encontrada en la salida estándar.

lookbib

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 bibliográficos en 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

mmroff es un preprocesador simple para groff.

neqn

El guión neqn formatea ecuaciones para salida ASCII.

nroff

El guión nroff emula al comando UNIX nroff usando groff.

pfbtops

pfbtops transforma una fuente en formato .pfb de Postscript a ASCII.

pic

pic compila descripciones de gráficos embebidos dentro de ficheros de entrada de troff o TeX a comandos que puedan ser entendidos por TeX o troff.

pre−grohtml y post−grohtml

pre− y post−grohtml transforman la salida de GNU troff a html.

refer

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

soelim lee ficheros y reemplaza líneas de la forma *.so fichero* por el contenido de *fichero*.

tbl

tbl compila descripciones de tablas embebidas dentro de ficheros de entrada troff a comandos que puedan ser entendidos por troff.

tfmtodit

tfmtodit crea un fichero de fuentes para su uso con **`groff -Tdvi`**.

troff

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.

zsoelim

zsoelim es la implementación GNU de soelim.

Dependencias de instalación de Groff

Última versión comprobada: 1.17.2.

Bash: sh
Binutils: ar, as, ld, ranlib
Bison: bison
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, touch
Gcc: cc1, cc1plus, collect2, cpp0, g++, gcc
Grep: egrep, grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, date, echo, expr, hostname, uname
Textutils: cat, tr

Gzip

Localización oficial para descarga

Gzip (1.2.4a):
ftp://ftp.gnu.org/gnu/gzip/

Parche para Gzip (1.2.4b):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Gzip

Última versión comprobada: 1.2.4a.

El paquete Gzip contiene programas para comprimir y descomprimir ficheros usando el codificador Lempel–Ziv (LZ77).

Gzip instala lo Siguiente:

Programas

gunzip (enlace a gzip), gzexe, gzip, uncompress (enlace a gunzip), zcat (enlace a gzip), zcmp, zdiff, zforce, zgrep, zmore y znew

Descripciones

Última versión comprobada: 1.2.4a.

Descripción de los programas

gunzip, uncompress

gunzip y uncompress descomprimen ficheros que hayan sido comprimidos con gzip.

gzexe

gzexe permite comprimir ficheros ejecutables que automáticamente se descomprimen y ejecutan al ser lanzados (con una penalización en el rendimiento).

gzip

gzip reduce el tamaño de los ficheros usando codificación Lempel–Ziv (LZ77).

zcat

zcat descomprime, y escribe en la salida estándar, tanto una lista de ficheros en su línea de comandos como un fichero leído por su entrada estándar.

zcmp

zcmp invoca al programa cmp en ficheros comprimidos.

zdiff

zdiff invoca al programa diff en ficheros comprimidos.

zforce

zforce fuerza la extensión .gz en todos los ficheros gzip 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

zgrep invoca al programa grep en ficheros comprimidos.

zmore

zmore es un filtro que permite examinar ficheros comprimidos de texto plano pantalla a pantalla en un terminal emulado (similar al programa more).

znew

znew recomprime ficheros en formato .Z (compress) al formato .gz (gzip).

Dependencias de instalación de Gzip

Última versión comprobada: 1.2.4a.

Bash: sh
Binutils: as, ld, nm
Fileutils: chmod, cp, install, ln, mv, rm
Gcc: cc1, collect2, cpp, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: hostname
Textutils: cat, tr

Kbd

Localización oficial para descarga

Kbd (1.08):
ftp://ftp.win.tue.nl/pub/linux−local/utils/kbd/

Parche para Kbd (1.08):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Kbd

Última versión comprobada: 1.06.

Kbd contiene ficheros de mapas de teclado y utilidades para el teclado.

Kbd instala lo siguiente:

Programas

chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, getunimap, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (enlace a psfxtable), psfgettable (enlace a psfxtable), psfstriptable (enlace a psfxtable), psfxtable, resizecons, setfont, setkeycodes, setleds, setlogcons, setmetamode, setvesablank, showfont, showkey, unicode_start, y unicode_stop

Descripciones

Última versión comprobada: 1.06.

Descripción de los programas

chvt

chvt cambia la terminal virtual que aparece en primer plano.

deallocvt

deallocvt desasigna las terminales virtuales no usadas.

dumpkeys

dumpkeys vuelca las tablas de traducción del teclado.

fgconsole

fgconsole muestra el número del terminal virtual activo.

getkeycodes

getkeycodes muestra la tabla de correspondencias de código de exploración (scan code) a código de teclas del núcleo.

getunimap

getunimap muestra el mapa unicode actualmente usado.

kbd_mode

kbd_mode muestra o establece el modo del teclado.

kbdrate

kbdrate establece la repetición y retardo del teclado.

loadkeys

loadkeys carga las tablas de traducción del teclado.

loadunimap

loadunimap carga la tabla de correspondencia de unicode a fuente del núcleo.

mapscrn

mapscrn carga una tabla de correspondencia de caracteres de salida, definida por el usuario, en el controlador de la consola. Este comando está obsoleto y sus funciones se encuentran incluidas en setfont.

openvt

openvt comienza un programa en un nuevo terminal virtual (VT).

psfaddtable, psfgettable, psfstriptable, psfxtable

Este es un grupo de herramientas para obtener tablas de caracteres Unicode a partir de fuentes de consola.

resizecons

resizecons cambia la idea del núcleo sobre el tamaño de la consola.

setfont

Esto permite cambiar las fuentes EGA/VGA de la consola.

setkeycodes

setkeycodes carga las entradas de la tabla de correspondencia de código de exploración (scan code) a código de tecla del núcleo.

setleds

setleds establece los LEDs del teclado. Mucha gente encuentra útil tener el bloqueo numérico (numlock) activado por defecto y usando este programa puede conseguirse.

setlogcons

setlogcons envía los mensajes del núcleo a la consola.

setmetamode

setmetamode define cómo se manejan las teclas meta del teclado.

setvesablank

Esto permite afinar el salvapantallas incorporado en el hardware (no animados, sólo una pantalla en blanco).

showfont

showfont muestra los datos de una fuente. La información mostrada incluye información de la fuente, sus propiedades, la métrica de los caracteres y el mapa de bits de cada carácter.

showkey

showkey examina los códigos de exploración (scan codes) y los códigos de tecla enviados por el teclado.

unicode_start

unicode_start pone la consola en modo Unicode.

unicode_stop

unicode_stop revierte el teclado y la consola del modo Unicode.

Dependencias de instalación de Kbd

Última versión comprobada: 1.06.

Bash: sh
Binutils: as, ld, strip
Bison: bison
Diffutils: cmp
Fileutils: cp, install, ln, mv, rm
Flex: flex
Gettext: msgfmt, xgettext
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Gzip: gunzip, gzip
Make: make
Patch: patch
Sed: sed
Sh−utils: uname

Less

Localización oficial para descarga

Less (378):
ftp://ftp.gnu.org/gnu/less/

Contenido de Less

Última versión comprobada: 374.

Less es un paginador de ficheros, o visor de texto. Muestra el contenido de un fichero, o de un flujo de datos, y tiene la habilidad de poder recorrerlo. Less tiene várias características no incluidas en el paginador **more**, como la habilidad de recorrer los ficheros hacia atrás.

Less instala lo siguiente:

Programas

less, lessecho y lesskey

Descripciones

Última versión comprobada: 374.

Descripción de los programas

less

El programa less es un paginador de ficheros (o visor de texto). Muestra el contenido de un fichero con la posibilidad de recorrerlo. Less es una evolución del paginador habitual llamado "more". Less tiene la habilidad de poder moverse a través de los ficheros y no necesita leer por completo el fichero al principio, lo que le hace rápido cuando se leen ficheros largos.

lessecho

lessecho es necesario para expandir meta−caracteres, como * y ?, en los nombres de ficheros en sistemas Unix.

lesskey

lesskey se usa para especificar los códigos de teclas usados por less.

Dependencias de instalación de Less

Última versión comprobada: 358.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, mv, rm, touch

Grep: egrep, grep
Gcc: cc1, collect2, cpp0, gcc
Make: make
Sed: sed
Sh−utils: expr, hostname, uname
Textutils: cat, tr

LFS−Bootscripts

Localización oficial para descarga

LFS−Bootscripts (1.11):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de LFS−bootscripts

Última versión comprobada: 1.11.

El paquete LFS−Bootscripts contiene guiones del interprete de comandos para el inicio del sistema al estilo SysV. Estos guiones realizan varias tareas como comprobar la integridad del sistema de ficheros durante el arranque, cargar el mapa del teclado, establecer la red y parar los procesos durante el cierre del sistema.

LFS−bootscripts instala lo siguiente:

Guiones

checkfs, cleanfs, functions, halt, ifdown, ifup, loadkeys, localnet, mountfs, mountproc, network, rc, reboot, sendsignals, setclock, swap, sysklogd y template

Descripciones

Última versión comprobada: 1.11.

Descripción de los guiones

checkfs

El guión checkfs comprueba los sistemas de ficheros justo antes de ser montados (con la excepción de los que usan registros de transacciones [journal] o los que se montan desde la red).

cleanfs

El guión cleanfs elimina los ficheros que no deben guardarse cuando se arranca de nuevo el sistema, como /var/run/* y /var/lock/*. Regenera /var/run/utmp y elimina los ficheros /etc/nologin, /fastboot y /forcefsck si existen.

functions

El guión functions contiene funciones usadas por diferentes guiones: chequeo de errores, chequeo de estado, etc.

halt

El guión halt se encarga de cerrar el sistema.

ifdown, ifup

Los guiones ifdown e ifup ayudan al guión network con los dispositivos de red.

loadkeys

El guión loadkeys carga el mapa que especifiques como apropiado para tu modelo de teclado.

localnet

El guión localnet establece el nombre de máquina usado por el sistema (hostname) y activa el dispositivo de red "loopback".

mountfs

El guión mountfs monta todos los sistemas de ficheros que no estén marcados como "noauto" o que no se monten a través de la red.

mountproc

El guión mountproc se usa para montar el sistema de ficheros proc.

network

El guión 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 guión rc es el controlador maestro de los niveles de arranque. Es el responsable de lanzar todos los demás guiones, uno a uno, en una secuencia específica.

reboot

El guión reboot se encarga de reiniciar el sistema.

sendsignals

El guión sendsignals se asegura de que todos los procesos terminen antes de parar o reiniciar el sistema.

setclock

El guión setclock fija el reloj del núcleo a la hora local en caso de que el reloj del ordenador no esté fijado a la hora GMT.

swap

El guión swap activa y desactiva las particiones y ficheros de intercambio (swap).

sysklogd

El guión sysklogd lanza y detiene los demonios de registro de eventos del sistema y del núcleo (syslogd y klogd).

template

El guión template es una plantilla para crear guiones de arranque personales para otros demonios y procesos.

Dependencias de instalación de LFS–Bootscripts

Última versión comprobada: 1.11.

Fileutils: chown, cp

Libtool

Localización oficial para descarga

Libtool (1.4.3):
ftp://ftp.gnu.org/gnu/libtool/

Contenido de Libtool

Última versión comprobada: 1.4.2.

GNU libtool es un guión para soporte genérico de librerías. Libtool oculta la complejidad del uso de librerías compartidas tras una interfaz consistente y portable.

Libtool instala lo siguiente:

Programas

libtool y libtoolize

Librerías

libltdl.a, libltdl.so (enlace a libltdl.so.3.1.0), libltdl.so.3 (enlace a libltdl.so.3.1.0) y libltdl.so.3.1.0

Descripciones

Última versión comprobada: 1.4.2.

Descripción de los programas

libtool

Libtool proporciona servicios de soporte generalizados para la compilación de librerías.

libtoolize

libtoolize proporciona una forma estándar de añadir soporte para libtool a un paquete.

Descripción de las librerías

libltdl, libltdl.so.3, libltdl.so.3.1.0

Una pequeña librería cuyo objetivo es ocultar las dificultades en la carga dinámica de librerías a los programadores.

Dependencias de instalación de Libtool

Última versión comprobada: 1.4.2.

Bash: sh
Binutils: ar, as, ld, nm, ranlib, strip
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gcc: cc, cc1, collect2, cpp0
Glibc: ldconfig
Grep: egrep, fgrep, grep
Make: make
Sed: sed
Sh−utils: echo, expr, hostname, sleep, uname
Texinfo: install−info
Textutils: cat, sort, tr, uniq

Lilo

Localización oficial para descarga

Lilo (22.2):
ftp://ibiblio.org/pub/Linux/system/boot/lilo/
http://ibiblio.org/pub/Linux/system/boot/lilo/

Contenido de Lilo

Última versión comprobada: 22.2.

Lilo es el LInux LOader, cargador de Linux.

Lilo instala lo siguiente:

Programas

lilo, mkrescue y keytab−lilo.pl

Descripciones

Última versión comprobada: 22.2.

Descripción de los programas

lilo

lilo instala el gestor de arranque que se usa para iniciar un sistema Linux.

mkrescue

mkrescue crea un disquete de rescate arrancable usando el núcleo existente y cualquier disco RAM de inicio.

keytab−lilo.pl

keytab−lilo.pl compila definiciones de teclado en un formato que lilo puede usar para ajustar el tipo de teclado durante el arranque.

Dependencias de instalación de Lilo

Última versión comprobada: 22.1.

Bash: sh
Bin86: as86, ld86
Binutils: as, ld, strip
Fileutils: cp, dd, ln
Gcc: cc, cc1, collect2, cpp0
Make: make
Sed: sed
Textutils: cat

Linux (el núcleo)

Localización oficial para descarga

Linux (2.4.20):
ftp://ftp.kernel.org/pub/linux/kernel/

Contenido de Linux

Última versión comprobada: 2.4.18.

El núcleo Linux es el corazón de todo sistema Linux. Es lo que hace a Linux funcionar. Cuando se enciende un ordenador y se inicia un sistema Linux, el núcleo es lo primero que se carga. El núcleo inicializa los componentes hardware del sistema: puertos serie, puertos paralelo, tarjetas de sonido, tarjetas de red, controladores IDE, controladores SCSI y mucho más. En pocas palabras, el núcleo hace que el hardware esté disponible para que el software pueda ejecutarse.

Linux instala lo siguiente:

Programas

El núcleo y las cabeceras del núcleo

Descripciones

Última versión comprobada: 2.4.18.

Descripción de los programas

Núcleo linux

El núcleo Linux es el corazón de todo sistema Linux. Es lo que hace a Linux funcionar. Cuando se enciende un ordenador y se inicia un sistema Linux, el núcleo es lo primero que se carga. El núcleo inicializa los componentes hardware del sistema: puertos serie, puertos paralelo, tarjetas de sonido, tarjetas de red, controladores IDE, controladores SCSI y mucho más. En pocas palabras, el núcleo hace que el hardware esté disponible para que el software pueda ejecutarse.

Ficheros de cabecera del núcleo linux

Estos son los ficheros que copiamos a `/usr/include/{linux,asm}` en el Capítulo 6. Deben coincidir con la versión con la que glibc ha sido compilada. *No* deben reemplazarse cuando se actualiza el núcleo. Son esenciales para compilar muchos programas.

Dependencias de instalación de Linux

Última versión comprobada: 2.4.17.

Bash: sh
Binutils: ar, as, ld, nm, objcopy
Fileutils: cp, ln, mkdir, mv, rm, touch
Findutils: find, xargs
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Gzip: gzip
Make: make
Gawk: awk
Modutils: depmod, genksyms
Net−tools: dnsdomainname, hostname
Sed: sed
Sh−utils: basename, date, expr, pwd, stty, uname, whoami, yes
Textutils: cat, md5sum, sort, tail

M4

Localización oficial para descarga

M4 (1.4):
ftp://ftp.gnu.org/gnu/m4/

Contenido de M4

Última versión comprobada 1.4.

M4 es un procesador de macros. Copia la entrada a la salida expandiendo las macros en el proceso. Las macros pueden ser internas o definidas por el usuario y pueden tomar cualquier número de argumentos. Aparte de hacer la expansión de macros, m4 tiene funciones internas para la inclusión de 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.

M4 instala lo siguiente:

Programas

m4

Descripciones

Última versión comprobada: 1.4.

Descripción del programa

m4

m4 es un procesador de macros. Copia la entrada a la salida expandiendo las macros en el proceso. Las macros pueden ser internas o definidas por el usuario y pueden tomar cualquier número de argumentos. Aparte de hacer la expansión de macros, m4 tiene funciones internas para la inclusión de 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.

Dependencias de instalación de M4

Última versión comprobada: 1.4.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, mv, rm
Make: make
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Sed: sed
Sh−utils: date, echo, hostname
Textutils: cat, tr

Make

Localización oficial para descarga

Make (3.80):
ftp://ftp.gnu.org/gnu/make/

Contenido de Make

Última versión comprobada: 3.79.1.

Make determina, automáticamente, qué piezas de un programa largo es necesario recompilar y ejecuta los comandos para recompilarlas.

Make instala lo siguiente:

Programas

make

Descripciones

Última versión comprobada: 3.79.1.

Descripción del programa

make

make determina, automáticamente, qué partes de un programa necesitan ser recompiladas, y lanza los comandos para hacerlo.

Dependencias de instalación de Make

Última versión comprobada: 3.79.1.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chgrp, chmod, install, ls, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

MAKEDEV

Localización oficial para descarga

MAKEDEV (1.7):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de MAKEDEV

Última versión comprobada: 1.5.

MAKEDEV es un guión que crea los ficheros de dispositivos estáticos necesarios, que usualmente residen en el directorio `/dev`. Puede encontrarse más información sobre los ficheros de dispositivos dentro de las fuentes del núcleo en `Documentation/devices.txt`.

MAKEDEV instala lo siguiente:

Programas

MAKEDEV

Descripciones

Última versión comprobada: 1.5.

Descripción del programa

MAKEDEV

`MAKEDEV` es un guión que crea los ficheros de dispositivos estáticos necesarios, que usualmente residen en el directorio `/dev`. Puede encontrarse más información sobre los ficheros de dispositivos dentro de las fuentes del núcleo en `Documentation/devices.txt`.

Dependencias de instalación de MAKEDEV

Última versión comprobada: 1.5.

Bash: sh
Fileutils: chmod, chown, cp, ln, mknod, mv, rm
Grep: grep
Sh−utils: expr, id

Man

Localización oficial para descarga

Man (1.5k):
ftp://ftp.win.tue.nl/pub/linux−local/utils/man/

Parche 80Cols para Man (1.5k):

http://downloads.linuxfromscratch.org/

Parche Manpath para Man (1.5k):
http://downloads.linuxfromscratch.org/

Parche Pager para Man (1.5k):
http://downloads.linuxfromscratch.org/

Contenido de Man

Última versión comprobada: 1.5k.

Man es un paginador de manuales.

Man instala lo siguiente:

Programas

apropos, makewhatis, man, man2dvi, man2html y whatis

Descripciones

Última versión comprobada: 1.5k.

Descripción de los programas

apropos

apropos busca palabras claves en un grupo de ficheros de bases de datos que contienen descripciones cortas de los comandos del sistema, y muestra el resultado en la salida estándar.

makewhatis

makewhatis lee todas las páginas de manual contenidas en las secciones incluidas en las rutas "manpath" o las páginas preformateadas contenidas en las secciones de las rutas "catpath". Por cada página escribe una línea en la base datos de whatis. Cada línea consiste en el nombre de la página y una descripción corta, separados por un guión. La descripción se extrae del contenido de la sección NAME de la página de manual.

man

man formatea y muestra las páginas de manual.

man2dvi

man2dvi convierte una página de manual a formato dvi.

man2html

man2html convierte una página de manual en html.

whatis

whatis busca palabras claves en un grupo de ficheros de bases de datos que contienen descripciones cortas de los comandos del sistema, y muestra el resultado en la salida estándar. Sólo las coincidencias de palabras completas son mostradas.

Dependencias de instalación de Man

Última versión comprobada: 1.5i.

Bash: sh
Binutils: as, ld
Fileutils: chmod, cp, install, mkdir, rm
Gcc: c11, collect2, cpp0, gcc
Grep: grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: echo
Textutils: cat

Man−pages

Localización oficial para descarga

Man−pages (1.54):
ftp://ftp.kernel.org/pub/linux/docs/manpages/

Contenido de Man−pages

Última versión comprobada: 1.54.

El paquete Man−pages contiene alrededor de 1200 páginas de manual. Esta documentación detalla las funciones de C y C++, describe varios ficheros de dispositivo importantes y proporciona documentación procedente de otros paquetes que posiblemente falte en los mismos.

Man−pages instala lo siguiente:

Ficheros de soporte

Varias páginas de manual.

Descripciones

Última versión comprobada: 1.54.

Descripción de los ficheros

páginas de manual

Un ejemplo de las páginas de manual incluidas son las que describen todas las funciones C y C++, algunos ficheros de /dev y otras cosas.

Dependencias de instalación de Man−pages

Última versión comprobada: 1.47.

Bash: sh
Fileutils: install
Make: make

Modutils

Localización oficial para descarga

Modutils (2.4.22):
ftp://ftp.kernel.org/pub/linux/utils/kernel/modutils/

Contenido de Modutils

Última versión comprobada: 2.4.16.

El paque Modutils contiene programas que puedes utilizar para trabajar con los módulos del núcleo.

Modutils instala lo siguiente:

Programas

depmod, genksyms, insmod, insmod_ksymoops_clean, kallsyms (enlace a insmod), kernelversion, ksyms (enlace a insmod), lsmod (enlace a insmod), modinfo, modprobe (enlace a insmod) y rmmod (enlace a insmod)

Descripciones

Última versión comprobada: 2.4.16.

Descripción de los programas

depmod

depmod maneja las descripciones de las dependencias para los módulos del núcleo

genksyms

genksyms lee (de la entrada estándar) la salida de gcc –E source.c y genera un fichero que contiene información sobre la versión.

insmod

insmod instala un módulo dentro del núcleo en ejecución.

insmod_ksymoops_clean

insmod_ksymoops_clean borra los símbolos del núcleo (ksyms) guardados y los módulos a los que no se ha accedido en los últimos 2 días.

kallsyms

kallsyms extrae todos los símbolos del núcleo para la depuración.

kernelversion

kernelversion informa sobre la versión mayor del núcleo en ejecución.

ksyms

ksyms muestra los símbolos exportados del núcleo.

lsmod

lsmod muestra información sobre todos los módulos cargados.

modinfo

modinfo examina un fichero objeto asociado con un módulo del núcleo y muestra la información que pueda encontrar.

modprobe

modprobe usa un fichero de dependencias similar a un Makefile creado por depmod, para cargar automáticamente el módulo o módulos necesarios del conjunto de módulos disponibles en el árbol de directorios predefinido.

rmmod

rmmod descarga los módulos del núcleo en ejecución.

Dependencias de instalación de Modutils

Última versión comprobada: 2.4.12.

Bash: sh
Binutils: ar, as, ld, ranlib, strip
Bison: bison
Diffutils: cmp
Fileutils: chmod, install, ln, mkdir, mv, rm
Flex: flex
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: basename, expr, hostname, uname
Textutils: cat, tr

Ncurses

Localización oficial para descarga

Ncurses (5.3):
ftp://ftp.gnu.org/gnu/ncurses/

Contenido de Ncurses

Última versión comprobada: 5.2.

El paquete Ncurses proporciona librerías para el manejo de caractéres y terminales, incluidos paneles y menús.

Ncurses instala lo siguiente:

Programas

captoinfo (enlace a tic), clear, infocmp, infotocap (enlace a tic), reset (enlace a tset), tack, tic, toe, tput y tset.

Librerías

libcurses.[a,so] (enlace a libncurses.[a,so]), libform.[a,so], libform_g.a, libmenu.[a,so], libmenu_g.a, libncurses++.a, libncurses.[a,so], libncurses_g.a, libpanel.[a,so] y libpanel_g.a

Descripciones

Última versión comprobada: 5.2.

Descripción de los programas

captoinfo

captoinfo convierte una descripción de termcap en una descripción de terminfo.

clear

clear limpia la pantalla si es posible. Busca en el entorno el tipo de terminal, y después en la base de datos de terminfo, para averiguar cómo limpiar la pantalla.

infocmp

infocmp puede usarse para comparar una entrada binaria de terminfo con otras entradas terminfo, reescribir una descripción de terminfo para aprovechar el campo "use=", o mostrar una descripción terminfo del fichero binario (term) en una variedad de formatos (lo opuesto de lo que hace tic).

infotocap

infotocap convierte una descripción terminfo en una descripción termcap.

reset

reset activa los modos "cooked" y "echo", quita los modos "cbreak" y "raw", activa la traslación de nueva línea y restablece cualquier carácter especial desactivado a sus valores por defecto, antes de hacer la inicialización del terminal de la misma manera que tset.

tack

tack es el comprobador de acciones de terminfo.

tic

tic es el compilador de entradas de descripciones de terminfo. El programa transforma un fichero terminfo en formato fuente a formato binario para su uso con las rutinas de las librerías ncurses. Los ficheros terminfo contienen información sobre las capacidades de un terminal.

toe

toe lista todos los tipos de terminal disponibles por su nombre primario, con descripciones.

tput

tput usa la base de datos de terminfo para poner a disposición del intérprete de comandos la información sobre las capacidades dependientes del terminal, para inicializar o restablecer el terminal, o para devolver el nombre largo del tipo de terminal requerido.

tset

tset inicializa los terminales para poder usarlos, pero no se usa posteriormente. Se incluye por compatibilidad con 4.4BSD.

Descripción de las librerías

libcurses, libncurses++, libncurses, libncurses_g

Estas librerías son la base del sistema y se usan para mostrar texto (a menudo de forma vistosa) en la pantalla. Un ejemplo donde se usa ncurses es en el proceso "make menuconfig" del núcleo.

libform, libform_g

libform se usa para implementar formularios en ncurses.

libmenu, libmenu_g

libmenu se usa para implementar menús en ncurses.

libpanel, libpanel_g

libpanel se usa para implementar paneles en ncurses.

Dependencias de instalación de Ncurses

Última versión comprobada: 5.2.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, mkdir, mv, rm
Gcc: c++, cc1, cc1plus, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, fgrep, grep
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, date, echo, expr, hostname, uname
Textutils: cat, sort, tr, wc

Netkit−base

Localización oficial para descarga

Netkit−base (0.17):
ftp://ftp.uk.linux.org/pub/linux/Networking/netkit/

Contenido de Netkit−base

Última versión comprobada: 0.17.

El paquete Netkit−base contiene dos herramientas de red: ping, que determina el tiempo de respuesta de los paquetes ICMP, y el demonio inetd, que escucha las peticiones a los puertos y asigna estas peticiones a los programas adecuados.

Netkit−base instala lo siguiente:

Programas

inetd y ping

Descripciones

Última versión comprobada: 0.17.

Descripción de los programas

inetd

inetd es la madre de todos los demonios (daemons). Escucha las peticiones de conexión y transfiere la llamada al demonio apropiado.

ping

ping envía paquetes ICMP ECHO_REQUEST a otra máquina y determina su tiempo de respuesta.

Dependencias de instalación de Netkit−base

Última versión comprobada: 0.17.

Bash: sh
Binutils: as, ld, strip
Fileutils: cp, install, rm
Make: make
Gcc: cc1, collect2, cpp0, gcc
Sed: sed
Sh−utils: date
Textutils: cat

Net−tools

Localización oficial para descarga

Net−tools (1.60):
http://www.tazenda.demon.co.uk/phil/net−tools/

Contenido de Net−tools

Última versión comprobada: 1.60.

El paquete Net−tools contiene una colección de programas que forman la base del trabajo en red en Linux.

Net−tools instala lo siguiente:

Programas

arp, dnsdomainname (enlace a hostname), domainname (enlace a hostname), hostname, ifconfig, nameif, netstat, nisdomainname (enlace a hostname), plipconfig, rarp, route, slattach y ypdomainname (enlace a hostname)

Descripciones

Última versión comprobada: 1.60.

Descripción de los programas

arp

arp se usa para manipular la caché ARP del núcleo, usualmente para añadir o borrar una entrada o volcar dicha caché.

dnsdomainname

dnsdomainname muestra el nombre del dominio DNS del sistema.

domainname

domainname muestra o establece el nombre del dominio NIS/YP del sistema.

hostname

hostname muestra o establece el nombre del sistema actual.

ifconfig

El comando ifconfig es el comando general usado para configurar las interfaces de red.

nameif

nameif nombra interfaces de red basándose en direcciones MAC.

netstat

netstat es una herramienta multipropósito usada para mostrar las conexiones de red, tablas de encaminamiento, estadísticas de las interfaces, conexiones enmascaradas y los miembros de conexiones multidestino (multicast).

nisdomainname

nisdomainname muestra o establece el nombre de dominio NIS/YP del sistema.

plipconfig

plipconfig se usa para afinar los parámetros del dispositivo PLIP, para hacerlo más rápido.

rarp

Relacionado con el programa arp, el programa rarp manipula la tabla RARP del sistema.

route

route es la utilidad general que se usa para manipular la tabla de encaminamiento IP.

slattach

slattach conecta una interfaz de red a una línea serie, esto es, pone una línea de terminal normal en uno o varios modos "de red".

ypdomainname

ypdomainname muestra o establece el nombre de dominio NIS/YP del sistema.

Dependencias de instalación de Net−tools

Última versión comprobada: 1.60.

Bash: bash, sh
Binutils: ar, as, ld
Fileutils: install, ln, ls, mv, rm
Gcc: cc, cc1, collect2, cpp0
Make: make
Sh−utils: echo

Patch

Localización oficial para descarga

Patch (2.5.4):
ftp://ftp.gnu.org/gnu/patch/

Contenido dePatch

Última versión comprobada: 2.5.4.

El programa patch modifica un fichero basandose en un parche. Normalmente un parche es una lista, creada por el programa diff, que contiene instrucciones sobre cómo debe modificarse el fichero original.

Patch instala lo siguiente:

Programas

patch

Descripciones

Última versión comprobada: 2.5.4.

Descripción del programa

patch

El programa patch modifica un fichero según lo indicado en un fichero de parche. Normalmente un fichero de parche es una lista, creada por el programa diff, que contiene instrucciones sobre cómo necesita ser modificado un fichero original. Patch se usa mucho para parchear el código fuente pues ahorra bastante tiempo y espacio. Imagina un paquete de 1MB de tamaño. La siguiente versión de ese paquete sólo cambia en dos ficheros con respecto a la primera versión. Se puede distribuir como un nuevo paquete entero de 1MB o sólo como un parche de 1KB con el que actualizar la primera versión para hacerla idéntica a la segunda. Por tanto, si la primera versión ya fue descargada, un parche evita hacer una segunda descarga larga.

Dependencias de instalación de Patch

Última versión comprobada: 2.5.4.

Bash: sh
Binutils: as, ld
Diffutils: cmp

Fileutils: chmod, install, mv, rm
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, grep
Make: make
Sed: sed
Sh−utils: echo, expr, hostname, uname
Textutils: cat, tr

Perl

Localización oficial para descarga

Perl (5.8.0):
http://www.perl.com/

Contenido de Perl

Última versión comprobada: 5.6.1.

El paquete Perl contiene perl, el Lenguaje Práctico de Extracción e Informe. Perl combina alguna de las mejores características de C, sed, awk y sh dentro de un poderoso lenguaje.

Perl instala lo siguiente:

Programas

a2p, c2ph, dprofpp, find2perl, h2ph, h2xs, perl, perl5.6.1, perlbug, perlcc, perldoc, pl2pm, pod2html, pod2latex, pod2man, pod2text, pod2usage, podchecker, podselect, pstruct, s2p y splain

Librerías

attrs.so, B.so, ByteLoader.so, DProf.so, Dumper.so, DynaLoader.a, Fcntl.so, Glob.so, Hostname.so, IO.so, libperl.a, Opcode.so, Peek.so, POSIX.so, re.so, SDBM_File.so, Socket.so, Syslog.so y SysV.so

Descripciones

Última versión comprobada: 5.6.1.

Descripción de los programas

a2p

a2p es un traductor de awk a perl.

c2ph

c2ph vuelca estructuras C similares a las generadas por "cc −g −S".

dprofpp

dprofpp muestra datos de perfiles perl.

find2perl

find2perl traduce líneas del comando find a código Perl.

h2ph

h2ph convierte ficheros de cabecera .h de C en ficheros de cabecera .ph de Perl.

h2xs

h2xs convierte ficheros de cabecera .h de C en extensiones de Perl.

perl, perl5.6.1

perl es el Lenguaje Práctico de Extracción e Informe. Combina algunas de las mejores características de C, sed, awk y sh en un poderoso lenguaje.

perlbug

perlbug ayuda en la generación de informes de errores sobre perl o sobre los módulos incorporados, y los envía por correo.

perlcc

perlcc genera ejecutables a partir de programas Perl.

perldoc

perldoc busca una parte de la documentación en formato .pod que se incluye en el árbol de instalación de perl o en un guión de perl, y lo muestra mediante "pod2man | nroff −man | $PAGER".

pl2pm

pl2pm es una herramienta que ayuda en la conversión de librerías .pl de estilo Perl4 en módulos de librería de estilo Perl5.

pod2html

pod2html convierte ficheros de formato pod a formato HTML.

pod2latex

pod2latex convierte ficheros de formato pod a formato LaTeX.

pod2man

pod2man convierte datos pod en entradas formateadas *roff.

pod2text

pod2text convierte datos pod en texto formateado ASCII.

pod2usage

pod2usage muestra mensajes de uso a partir de documentos pod incluidos en ficheros.

podchecker

podchecker chequea la sintaxis de los ficheros de documentación en formato pod.

podselect

podselect muestra las secciones seleccionadas de la documentación pod en la salida estándar.

pstruct

pstruct vuelca estructuras C similares a las generadas por "cc −g −S".

s2p

s2p es un traductor de sed a perl.

splain

splain es un programa que fuerza diagnósticos de avisos exhaustivos en perl.

Descripción de las librerías

attrs

No hay descripción disponible.

B

No hay descripción disponible.

ByteLoader

No hay descripción disponible.

DProf

No hay descripción disponible.

Dumper

No hay descripción disponible.

DynaLoader

No hay descripción disponible.

Fcntl

No hay descripción disponible.

Glob

No hay descripción disponible.

Hostname

No hay descripción disponible.

IO

No hay descripción disponible.

libperl

No hay descripción disponible.

Opcode

No hay descripción disponible.

Peek

No hay descripción disponible.

POSIX

No hay descripción disponible.

re

No hay descripción disponible.

SDBM_File

No hay descripción disponible.

Socket

No hay descripción disponible.

Syslog

No hay descripción disponible.

SysV

No hay descripción disponible.

Dependencias de instalación de Perl

Última versión comprobada: 5.6.1.

Bash: sh
Binutils: ar, as, ld, nm
Diffutils: cmp
Fileutils: chmod, cp, ln, ls, mkdir, mv, rm, touch
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Gawk: awk
Sed: sed
Sh−utils: basename, date, echo, expr, hostname, pwd, uname, whoami
Textutils: cat, comm, sort, split, tr, uniq, wc

Procinfo

Localización oficial para descarga

Procinfo (18):
ftp://ftp.cistron.nl/pub/people/svm/

Contenido de Procinfo

Última versión comprobada: 18.

El programa procinfo obtiene datos del sistema, como el uso de la memoria y los números de las interrupciones (IRQ), a partir del directorio `/proc`, y formatea estos datos de una forma atractiva.

Procinfo instala lo siguiente:

Programas

lsdev, procinfo y socklist

Descripciones

Última versión comprobada: 18.

Descripción de los programas

lsdev

lsdev recopila la información sobre los dispositivos físicos instalados en su ordenador a partir de los ficheros con las interrupciones, puertos de entrada/salida y acceso directo a memoria (DMA) del directorio /proc, facilitando una descripción rápida de qué direcciones de entrada/salida (I/O) y qué canales IRQ y DMA usa cada dispositivo.

procinfo

procinfo recopila algunos datos del sistema a partir del directorio /proc y los muestra en un bonito formato en el dispositivo de salida estándar.

socklist

Es un guión de Perl que facilita una lista de todos los conectores de red (sockets) abiertos, enumerando el tipo, puerto, inodo, identificador de usuario (uid), identificador de proceso (pid), descriptor de ficheros (fd) y el programa al que pertenece.

Dependencias de instalación de Procinfo

Última versión comprobada: 18.

Binutils: as, ld
Fileutils: install, mkdir
Gcc: cc1, collect2, cpp0, gcc
Make: make

Procps

Localización oficial para descarga

Procps (3.1.5):
http://procps.sourceforge.net/

Parche para Procps (3.1.5):
http://downloads.linuxfromscratch.org/

Contenido de Procps

Última versión comprobada: 2.0.7.

El paquete Procps proporciona programas para supervisar y parar procesos del sistema. Procps obtiene la información sobre los procesos a través del directorio `/proc`.

Procps instala lo siguiente:

Programas

free, kill, oldps, pgrep, pkill, ps, skill, snice, sysctl, tload, top, vmstat, w y watch

Librerías

libproc.so

Descripciones

Última versión comprobada: 2.0.7.

Descripción de los programas

free

free muestra la cantidad total de memoria física y de intercambio (swap) libre y usada en el sistema, al igual que la memoria compartida (shared) y de almacenamiento intermedio (buffers) usada por el núcleo.

kill

kill envía señales a los procesos.

oldps y ps

ps facilita una instantánea de los procesos actuales.

pgrep

pgrep visualiza procesos basándose en el nombre u otros atributos

pkill

pkill envía señales a procesos basándose en el nombre u otros atributos

skill

skill envía señales a procesos que coincidan con un criterio.

snice

snice cambia la prioridad de planificación de los procesos que coincidan con un criterio.

sysctl

sysctl modifica los parámetros del núcleo en tiempo de ejecución.

tload

tload imprime un gráfico de la carga actual del sistema en la consola (tty) especificada o, si no se especifica ninguna, la consola del proceso tload.

top

top proporciona una vista dinámica de la actividad del procesador en tiempo real.

vmstat

vmstat muestra información sobre los procesos, memoria, paginación, entrada/salida por bloques y actividad del procesador.

w

w muestra información sobre los usuarios que hay actualmente en el sistema y sus procesos.

watch

watch lanza comandos repetidamente, mostrando sus salidas (a pantalla completa la primera).

Descripción de la librería

libproc

libproc es la librería contra la que muchos de los programas de este grupo están enlazados, para ahorrar espacio en disco implementando las funciones comunes sólo una vez.

Dependencias de instalación de Procps

Última versión comprobada: 2.0.7.

Bash: sh
Binutils: as, ld, strip
Fileutils: install, ln, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: grep
Make: make
Gawk: awk
Sed: sed

Sh−utils: basename, pwd
Textutils: sort, tr

Psmisc

Localización oficial para descarga

Psmisc (21.2):
http://download.sourceforge.net/psmisc/
ftp://download.sourceforge.net/pub/sourceforge/psmisc/

Contenido de Psmisc

Última versión comprobada: 21.

El paquete Psmisc contiene tres programas que ayudan en el manejo del directorio `/proc`.

Psmisc instala lo siguiente:

Programas

fuser, killall y pstree

Descripciones

Última versión comprobada: 21.

Descripción de los programas

Nota: en LFS no instalamos el enlace pidof por omisión porque usamos en su lugar el programa pidof de sysvinit.

fuser

fuser muestra los números de identificación de los procesos (PID) que usan los ficheros o sistemas de ficheros especificados.

killall

killall envía una señal a todos los procesos que ejecutan alguno de los comandos especificados.

pstree

pstree muestra los procesos en ejecución en forma de árbol.

Dependencias de instalación de Psmisc

Última versión comprobada: 20.2

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Bison: bison
Binutils: as, ld
Diffutils: cmp
Fileutils: chmod, install, ls, mkdir, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Grep: egrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Texinfo: makeinfo
Textutils: cat, tr

Sed

Localización oficial para descarga

Sed (4.0.5):
ftp://ftp.gnu.org/gnu/sed/

Contenido de Sed

Última versión comprobada: 3.02.

sed es un editor de flujo. Un editor de flujo se utiliza para realizar transformaciones básicas de texto sobre un flujo de entrada (un fichero o la entrada procedente de una tubería).

Sed instala lo siguiente:

Programas

sed

Descripciones

Última versión comprobada: 3.02.

Descripción del programa

sed

sed es un editor de flujo. Un editor de flujo se usa para realizar transformaciones básicas de texto en un flujo de entrada (un fichero o una tubería).

Dependencias de instalación de Sed

Última versión comprobada: 3.02.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Sed: sed
Sh−utils: echo, expr, hostname, sleep
Texinfo: install−info, makeinfo
Textutils: cat, tr

Shadow

Localización oficial para descarga

Shadow (4.0.3):
ftp://ftp.pld.org.pl/software/shadow/

Contenido de Shadow

Última versión comprobada: 4.0.3.

El paquete Shadow se creó para consolidar la seguridad del sistema de contraseñas.

Shadow instala lo siguiente:

Programas

chage, chfn, chpasswd, chsh, dpasswd, expiry, faillog, gpasswd, groupadd, groupdel, groupmod, groups, grpck, grpconv, grpunconv, lastlog, login, logoutd, mkpasswd, newgrp, newusers, passwd, pwck, pwconv, pwunconv, sg (enlace a newgrp), useradd, userdel, usermod, vigr (enlace a vipw) y vipw

Librerías

libmisc y libshadow

Descripciones

Última versión comprobada: 4.0.3.

Descripción de los programas

chage

chage cambia el número de días entre cambios de la contraseña y la fecha del último cambio de contraseña.

chfn

chfn cambia el nombre completo de un usuario y otra información (extensión y número de teléfono de su oficina y número de teléfono de su casa).

chpasswd

chpasswd lee un fichero con pares de usuarios y contraseñas en la entrada estándar y usa esta información para actualizar un grupo de usuarios existentes.

chsh

chsh cambia el intérprete de comandos que se ejecuta cuando un usuario entra al sistema.

dpasswd

dpasswd añade, borra y actualiza las contraseñas de acceso telefónico del usuario.

expiry

Comprueba y fuerza la política de expiración de contraseñas.

faillog

faillog formatea el contenido del registro de fallos /var/log/faillog, y mantiene el contador y los límites de fallos.

gpasswd

gpasswd se usa para administrar el fichero /etc/group.

groupadd

El comando groupadd crea un nuevo grupo usando los valores especificados en la línea de comandos y los valores por defecto del sistema.

groupdel

El comando groupdel modifica los ficheros con las cuentas del sistema, borrando todas las entradas referidas a un determinado grupo.

groupmod

El comando groupmod modifica los ficheros de las cuentas del sistema para reflejar los cambios especificados en la línea de comandos.

groups

groups muestra los grupos a los que pertenece un usuario.

grpck

grpck verifica la integridad de la información de autenticación del sistema.

grpconv

grpunconv convierte a ficheros de grupos ocultos (shadow group files) los ficheros de grupos normales.

grpunconv

grpunconv convierte los ficheros de grupos ocultos en ficheros de grupos normales.

lastlog

lastlog formatea y muestra el contenido del registro de último acceso /var/log/lastlog. Muestra el nombre con el que se ha accedido, puerto y hora del último acceso.

login

login se usa para establecer una nueva sesión con el sistema.

logoutd

logoutd fuerza las restricciones de hora de acceso y puerto especificadas en /etc/porttime.

mkpasswd

mkpasswd lee un fichero en el formato facilitado por las opciones y lo convierte al formato de fichero de base de datos correspondiente.

newgrp

newgrp se usa para cambiar el identificador de grupo actual durante una sesión de acceso.

newusers

newusers lee un fichero con pares de nombres de usuario y contraseñas en texto plano y usa esa información para actualizar un grupo de usuarios existentes o para crear nuevos usuarios.

passwd

passwd cambia las contraseñas de las cuentas de usuarios y grupos.

pwck

pwck verifica la integridad de los ficheros que contienen las contraseñas.

pwconv

pwconv convierte a ficheros de contraseñas ocultas los ficheros de contraseñas normales.

pwunconv

pwunconv convierte los ficheros de contraseñas ocultas en ficheros normales.

sg

sg asigna el grupo del usuario al especificado, o ejecuta un comando como miembro del grupo indicado.

useradd

useradd crea un nuevo usuario o actualiza la información por defecto de un nuevo usuario.

userdel

userdel modifica los ficheros con las cuentas del sistema, borrando todas las entradas referidas al nombre de acceso especificado.

usermod

usermod modifica los ficheros con las cuentas del sistema para reflejar los cambios especificados en la línea de comandos.

vipw y vigr

vipw y vipgr pueden editar los ficheros /etc/passwd y /etc/group, respectivamente. Con la opción –s, pueden editar la versión oculta de dichos ficheros, /etc/shadow y /etc/gshadow, respectivamente.

Descripción de las librerías

libmisc

No hay descripción disponible.

libshadow

libshadow proporciona funcionalidades comunes para los programas del paquete shadow.

Dependencias de instalación de Shadow

Última versión comprobada: 20001016.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, nm, ranlib
Diffutils: cmp
Fileutils: chmod, cp, install, ln, ls, mkdir, mv, rm, rmdir
Gettext: msgfmt, xgettext
Gcc: cc1, collect2, cpp0, gcc
Glibc: ldconfig
Grep: egrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Texinfo: makeinfo
Textutils: cat, sort, tr, uniq

Sh−utils

Localización oficial para descarga

Sh−utils (2.0):
ftp://ftp.gnu.org/gnu/sh−utils/

Parche para Sh−utils (2.0):
http://downloads.linuxfromscratch.org/

Parche Hostname para Sh−utils (2.0−hostname):
http://downloads.linuxfromscratch.org/

Contenido de Sh−utils

Última versión comprobada: 2.0.

El paquete Sh−utils contiene un número de utilidades para realizar manipulaciones básicas en el intérprete de comandos.

Sh−utils instala lo siguiente:

Programas

basename, chroot, date, dirname, echo, env, expr, factor, false, groups, hostid, id, logname, nice, nohup, pathchk, pinky, printenv, printf, pwd, seq, sleep, stty, su, tee, test, true, tty, uname, uptime, users, who, whoami y yes

Descripciones

Última versión comprobada: 2.0.

Descripción de los programas

basename

basename elimina los directorios y las extensiones de los nombres de ficheros.

chroot

chroot ejecuta un comando o un intérprete de comandos (shell) interactivo dentro de un directorio raíz determinado.

date

date muestra la fecha y hora actual en un formato determinado o establece la fecha y hora del sistema.

dirname

dirname elimina los sufijos que no son directorios del nombre de un fichero.

echo

echo muestra una línea de texto.

env

env ejecuta un programa en un entorno modificado.

expr

expr evalúa expresiones.

factor

factor muestra los factores primos de los números enteros especificados.

false

false siempre termina con un código de estado que indica un fallo.

groups

groups muestra los grupos a los que pertenece un usuario.

hostid

hostid muestra el identificador numérico (en hexadecimal) de la máquina actual.

id

id muestra los identificadores efectivos de usuario y de grupo del usuario actual o de un usuario dado.

logname

logname muestra el nombre de acceso (login name) del usuario actual.

nice

nice ejecuta un programa con una prioridad distinta.

nohup

nohup ejecuta un comando que no se interrumpe cuando se cierra la sesión, con su salida a un fichero de registro.

pathchk

pathchk comprueba si los nombres de ficheros son válidos o portables.

pinky

pinky es una utilidad parecida a finger que obtiene información sobre un determinado usuario.

printenv

printenv muestra todo o parte del entorno.

printf

printf formatea y muestra datos (igual que la función printf de C).

pwd

pwd muestra el nombre del directorio de trabajo actual.

seq

seq muestra números en un cierto rango y con un cierto incremento.

sleep

sleep establece un retardo durante un determinado instante de tiempo.

stty

stty cambia y muestra las opciones de configuración del terminal.

su

su ejecuta un intérprete de comandos (shell) con un identificador de usuario y de grupo diferentes.

tee

tee lee de la entrada estándar y escribe en la salida estándar y en ficheros.

test

test comprueba el tipo de los ficheros y compara valores.

true

true siempre termina con un código de estado que indica éxito.

tty

tty muestra el nombre de fichero del terminal conectado a la entrada estándar.

uname

uname muestra información del sistema.

uptime

uptime muestra cuanto tiempo hace que el sistema está en marcha.

users

users muestra los nombres de los usuarios conectados actualmente.

who

who muestra quién está conectado.

whoami

whoami muestra el nombre de usuario asociado con el identificador de usuario efectivo actual.

yes

yes muestra en pantalla 'y' o una cadena de texto dada indefinidamente.

Dependencias de instalación de Sh−utils

Última versión comprobada: 2.0.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, chown, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tr

Sysklogd

Localización oficial para descarga

Sysklogd (1.4.1):
http://www.infodrom.org/projects/sysklogd/

Contenido de Sysklogd

Última versión comprobada: 1.4.1.

El paquete Sysklogd contiene programas para grabar tanto los mensajes del sistema como los generados por el núcleo

Sysklogd instala lo siguiente:

Programas

klogd y syslogd

Descripciones

Última versión comprobada: 1.4.1.

Descripción de los programas

klogd

klogd es un demonio del sistema que intercepta y registra los mensajes del núcleo Linux.

syslogd

syslogd proporciona una forma de registrar sucesos que muchos programas modernos utilizan. Cada mensaje registrado contiene como mínimo un campo con el nombre de la máquina y la fecha y, normalmente, también un campo con el nombre del programa. Pero eso depende de lo amigable que sea el programa a registrar.

Dependencias de instalación de Sysklogd

Última versión comprobada: 1.4.1.

Binutils: as, ld, strip
Fileutils: install
Gcc: cc1, collect2, cpp0, gcc
Make: make

Sysvinit

Localización oficial para descarga

Sysvinit (2.84):
ftp://ftp.cistron.nl/pub/people/miquels/sysvinit/

Contenido de Sysvinit

Última versión comprobada: 2.84.

El paquete Sysvinit contiene programas para controlar el arranque, ejecución y descarga de todos los demás programas.

Sysvinit instala lo siguiente:

Programas

halt, init, killall5, last, lastb (enlace a last), mesg, pidof (enlace a killall5), poweroff (enlace a halt), reboot (enlace a halt), runlevel, shutdown, sulogin, telinit (enlace a init), utmpdump y wall

Descripciones

Última versión comprobada: 2.84.

Descripción de los programas

halt

halt anota en el fichero /var/log/wtmp que el sistema se va a venir abajo, y entonces le indica al núcleo que cierre, reinicie o apague el sistema. Si halt o reboot son llamados cuando el sistema no está en los niveles de ejecución 0 o 6, shutdown es invocado en su lugar (con las opciones –h o –r).

init

init es el padre de todos los procesos. Su función principal es crear procesos a partir de un guión almacenado en el fichero /etc/inittab. Este fichero normalmente tiene unas entradas que hacen que init active la creación de los terminales en cada línea desde la que los usuarios pueden conectarse. También controla los procesos autónomos requeridos por un sistema particular.

killall5

killall5 es el comando killall de SystemV. Envía una señal a todos los procesos excepto a los procesos de su propia sesión, por tanto no puede matar el intérprete de comandos en el que se esté ejecutando el guión desde el que fue llamado.

last

last busca hacia atrás en el fichero /var/log/wtmp (o el fichero indicado con la opción –f) y muestra una lista con todos los usuarios conectados (y desconectados) desde que el fichero fue creado.

lastb

lastb es lo mismo que last, excepto que por defecto muestra los registros del fichero /var/log/btmp, que contiene todos los intentos fallidos de conexión.

mesg

mesg controla el acceso al terminal de un usuario por otros. Se usa para permitir o denegar a otros usuarios escribir en su terminal.

pidof

pidof muestra los identificadores de proceso (PIDs) de los programas especificados.

poweroff

poweroff es equivalente a "shutdown −h −p now". Para el ordenador y lo apaga (cuando se usa una BIOS compatible APM y APM está activado en el núcleo).

reboot

reboot es equivalente a "shutdown −r now". Reinicia el ordenador.

runlevel

runlevel lee el fichero utmp del sistema (usualmente /var/run/utmp) para localizar el registro del nivel de ejecución, y entonces muestra el anterior y el nivel de ejecución actual del sistema en la salida estándar, separados por un espacio.

shutdown

shutdown provoca la caída del sistema de una forma segura. Todos los usuarios conectados son notificados de que el sistema se va a venir abajo, y se bloquean los intentos de conexión al sistema.

sulogin

sulogin es invocado por init cuando el sistema entra en el modo monousuario (esto se hace mediante una entrada en /etc/inittab). Init también intenta ejecutar sulogin cuando se le pasa la opción −b desde el gestor de arranque (p.e., LILO).

telinit

telinit envía las señales apropiadas a init, diciendo en qué nivel de ejecución debe entrar.

utmpdump

utmpdump muestra el contenido de un fichero (usualmente /var/run/utmp) en la salida estándar en un formato comprensible por el usuario.

wall

wall envía un mensaje a todos los usuarios conectados que tengan los permisos de mesg puestos a "yes".

Dependencias de instalación de Sysvinit

Última versión comprobada: 2.84.

Bash: sh
Binutils: as, ld

Fileutils: chown, cp, install, ln, mknod, rm
Gcc: cc, cc1, collect2, cpp0
Make: make
Sed: sed

Tar

Localización oficial para descarga

Tar (1.13):
ftp://ftp.gnu.org/gnu/tar/

Parche para Tar (1.13):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Tar

Última versión comprobada: 1.13.

Tar es un programa de archivado diseñado para almacenar y estraer ficheros en un archivo conocido como fichero tar.

Tar instala lo siguiente:

Programas

rmt y tar

Descripciones

Última versión comprobada: 1.13.

Descripción de los programas

rmt

rmt es un programa utilizado, en modo remoto, por los programas dump y restore para manipular una unidad de cinta magnética mediante una comunicación de conexión entre procesos.

tar

tar es un programa de achivación diseñado para almacenar y extraer ficheros de un archivo conocido como fichero tar.

Dependencias de instalación de Tar

Última versión comprobada: 1.13.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Patch: patch
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Texinfo: install−info, makeinfo
Textutils: cat, tr

Texinfo

Localización oficial para descarga

Texinfo (4.3):
ftp://ftp.gnu.org/gnu/texinfo/

Contenido de Texinfo

Última versión comprobada: 4.2.

El paquete Texinfo contiene programas usados para leer, escribir y convertir documentos Info, que suministran documentación del sistema.

Texinfo instala lo siguiente:

Programas

info, infokey, install−info, makeinfo, texi2dvi y texindex

Descripciones

Última versión comprobada: 4.2.

Descripción de los programas

info

El programa info lee documentos Info, almacenados normalmente en el directorio /usr/share/info. Los documentos Info son como las páginas de manual, pero tienden a ser más profundos que una simple explicación de las opciones de un programa.

infokey

infokey compila un fichero fuente que contiene opciones de Info en un formato binario.

install–info

El programa install–info actualiza las entradas info. Cuando se ejecuta info, muestra una lista con los temas disponibles (es decir, los documentos info) disponibles. El programa install–info se usa para mantener esta lista. Si los ficheros info son eliminados manualmente, también debes eliminar el tema en el fichero índice. Este programa se utiliza para eso. También funciona en sentido contrario cuando se añaden documentos info.

makeinfo

El programa makeinfo convierte documentos fuente Texinfo a varios formatos. Los formatos disponibles son: ficheros info, texto plano y HTML.

texi2dvi

El programa texi2dvi imprime documentos Texinfo.

texindex

El programa texindex se usa para ordenar ficheros índice de Texinfo.

Dependencias de instalación de Texinfo

Última versión comprobada: 4.0.

Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ln, ls, mkdir, mv, rm
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, fgrep, grep

Make: make
Sed: sed
Sh−utils: basename, echo, expr, hostname, sleep
Texinfo: makeinfo
Textutils: cat, tr

Textutils

Localización oficial para descarga

Textutils (2.1):
ftp://ftp.gnu.org/gnu/textutils/

Contenido deTextutils

Última versión comprobada: 2.0.

El paquete Textutils contiene varios programas para procesar y manipular ficheros de texto.

Textutils instala lo siguiente:

Programas

cat, cksum, comm, csplit, cut, expand, fmt, fold, head, join, md5sum, nl, od, paste, pr, ptx, sort, split, sum, tac, tail, tr, tsort, unexpand, uniq y wc

Descripciones

Última versión comprobada: 2.0.

Descripción de los programas

cat

cat concatena ficheros o la entrada estándar en la salida estándar.

cksum

cksum muestra la suma de comprobación CRC y cuenta los bytes de un fichero.

comm

comm compara dos ficheros ordenados línea por línea.

csplit

csplit escribe partes de un fichero separadas por un determinado patrón en ficheros xx01, xx02, etc, y muestra el número de bytes de cada parte en la salida estándar.

cut

cut imprime en la salida estándar las partes seleccionadas de las líneas de un fichero.

expand

expand convierte las tabulaciones de un fichero en espacios, escribiendo en la salida estándar

fmt

fmt reformatea cada párrafo de un fichero, escribiendo en la salida estándar.

fold

fold reajusta la longitud de las líneas de un fichero (por defecto, la entrada estándar), escribiendo en la salida estándar.

head

head imprime en la salida estándar las primeras xx (10 por defecto) líneas de un fichero.

join

join une líneas de dos ficheros en un campo común.

md5sum

md5sum muestra o chequea sumas de comprobación MD5.

nl

nl escribe un fichero en la salida estándar, añadiendo números de línea.

od

od escribe en la salida estándar una representación inequívoca (por defecto en octal) de un fichero.

paste

paste escribe en la entrada estándar líneas formadas por las líneas de cada uno de los ficheros especificados, separadas por tabulaciones.

pr

pr pagina o muestra en columnas el texto de un fichero, para imprimirlo posteriormente.

ptx

ptx genera un índice permutado de los contenidos de un fichero.

sort

sort escribe en la salida estándar una concatenación de ficheros ordenados.

split

split divide un fichero en partes de tamaño fijo llamadas FICHEROaa, FICHERObb,...

sum

sum muestra la suma de comprobación y el número de bloques que ocupa un fichero.

tac

tac escribe un fichero o ficheros en la salida estándar, comenzando por la última línea.

tail

tail imprime en la salida estándar las últimas xx (10 por defecto) líneas de un fichero.

tr

tr convierte, altera y/o borra caracteres de la entrada estándar, escribiendo en la salida estándar.

tsort

tsort escribe una lista totalmente ordenada de acuerdo con el orden parcial de los ficheros especificados.

unexpand

unexpand convierte los espacios de un fichero en tabulaciones, escribiendo en la salida estándar.

uniq

uniq elimina las líneas duplicadas de un fichero ordenado.

wc

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.

Dependencias de instalación de Textutils

Última versión comprobada: 2.0.

Autoconf: autoconf, autoheader
Automake: aclocal, automake
Bash: sh
Binutils: ar, as, ld, ranlib
Diffutils: cmp
Fileutils: chmod, install, ls, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp0, gcc
Glibc: getconf
Grep: egrep, fgrep, grep
M4: m4
Make: make
Gawk: gawk
Net−tools: hostname
Perl: perl
Sed: sed
Sh−utils: basename, echo, expr, sleep, uname
Tar: tar
Texinfo: install−info, makeinfo
Textutils: cat, tr

Util−linux

Localización oficial para descarga

Util−linux (2.11y):
ftp://ftp.win.tue.nl/pub/linux−local/utils/util−linux/

Contenido de Util−linux

Última versión comprobada: 2.11t.

El paquete Util−linux contiene una miscelánea de utilidades. Alguna de estas utilidades más destacables se utilizan para montar, desmontar, formatear, particionar y manejar dispositivos de disco, abrir puertos de consola o capturar los mensajes del núcleo.

Util−linux instala lo siguiente:

Programas

agetty, arch, blockdev, cal, cfdisk, chkdupexe, col, colcrt, colrm, column, ctrlaltdel, cytune, ddate, dmesg, elvtune, fdformat, fdisk, 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, parse.bash, parse.tcsh, pg, pivot_root, ramsize (enlace a rdev), raw, rdev, readprofile, rename, renice, rev, rootflags (enlace a rdev), script, setfdprm, setsid, setterm, sfdisk, swapoff (enlace a swapon), swapon, test.bash, test.tcsh, tunelp, ul, umount, vidmode (enlace a rdev), whereis y write

Descripciones

Última versión comprobada: 2.11t.

Descripción de los programas

agetty

agetty abre un puerto de terminal, espera la introducción de un nombre de usuario e invoca al comando /bin/login.

arch

arch muestra la arquitectura de la máquina.

blockdev

blockdev permite llamar a los controles de entrada/salida (ioctls) de los dispositivos de bloque desde la línea de comandos.

cal

cal muestra un calendario simple.

cfdisk

cfdisk es un manipulador de la tabla de particiones del disco basado en libncurses.

chkdupexe

chkdupexe encuentra ejecutables duplicados.

col

col filtra avances de línea inversos de la entrada.

colcrt

colcrt filtra la salida de nroff para su visualización en un CRT.

colrm

colrm elimina columnas de un fichero.

column

column muestra listas en columnas.

ctrlaltdel

ctrlaltdel establece la función de la combinación de teclas CTRL+ALT+DEL (reinicio duro o blando).

cytune

cytune afina los parámetros del controlador de Cyclades.

ddate

ddate convierte las fechas Gregorianas en fechas Discordantes.

dmesg

dmesg se usa para examinar o controlar el anillo de almacenamiento intermedio del núcleo (los mensajes de arranque del núcleo).

elvtune

elvtune permite afinar el nivel de entrada/salida por cola de un dispositivo de bloques.

fdformat

fdformat formatea un disquete a bajo nivel.

fdisk

fdisk es un manejador de la tabla de particiones.

fsck.cramfs

No hay descripción disponible.

fsck.minix

fsck.minix realiza una comprobación de consistencia en sistemas de ficheros MINIX de Linux.

getopt

getopt analiza opciones de comandos de la misma forma que el comando getopt de C.

hexdump

hexdump muestra un fichero, o la entrada estándar, en un formato especificado por el usuario (ASCII, decimal, hexadecimal, octal).

hwclock

hwclock interroga y pone en hora el reloj del ordenador (también llamado RTC o reloj BIOS).

ipcrm

ipcrm elimina el recurso especificado.

ipcs

ipcs facilita información sobre los recursos IPC.

isosize

isosize muestra el tamaño de un sistema de ficheros iso9660.

line

line copia una línea (hasta el carácter de nueva línea) de la entrada estándar y la escribe en la salida estándar.

logger

logger crea entradas en el registro del sistema.

look

look muestra líneas que comienzan con una cadena dada.

losetup

losetup activa y controla los dispositivos de bucle (loop).

mcookie

mcookie genera galletas mágicas (magic cookies) para xauth.

mkfs

mkfs construye un sistema de ficheros Linux en un dispositivo, normalmente una partición del disco duro.

mkfs.bfs

mkfs.bfs crea un sistema de ficheros bfs de SCO en un dispositivo, normalmente una partición del disco duro.

mkfs.cramfs

No hay descripción disponible.

mkfs.minix

mkfs.minix crea un sistema de ficheros MINIX en un dispositivo, normalmente una partición del disco duro

mkswap

mkswap configura un área de intercambio (swap) de Linux en un dispositivo o en un fichero.

more

more es un filtro para paginar texto pantalla a pantalla.

mount

mount monta, a partir de muchas posibles fuentes, sistemas de ficheros o directorios en un directorio (punto de montaje).

namei

namei sigue el nombre de una ruta hasta encontrar el punto terminal.

parse.bash, parse.tcsh, test.bash, test.tcsh

Estos son guiones de ejemplo de uso del programa getopt con BASH o TCSH.

pg

No hay descripción disponible.

pivot_root

pivot_root cambia el sistema de ficheros raíz del proceso actual.

ramsize

ramsize muestra y establece el tamaño del disco RAM.

raw

raw se usa para unir un dispositivo Linux de carácter a un dispositivo de bloque.

rdev

rdev muestra y establece el dispositivo raíz de la imagen, el dispositivo de intercambio, el tamaño del disco RAM, o el modo de vídeo.

readprofile

readprofile lee la información de los perfiles del núcleo.

rename

rename renombra ficheros.

renice

renice altera la prioridad de los procesos en ejecución.

rev

rev invierte el orden de las líneas de un fichero.

rootflags

rootflags muestra y establece la información extra usada cuando se monta el sistema de ficheros raíz.

script

script registra todo lo que se ha tecleado en la sesión de terminal.

setfdprm

setfdprm establece los parámetros facilitados por el usuario para los disquetes.

setsid

setsid lanza programas en una nueva sesión.

setterm

setterm establece los parámetros del terminal.

sfdisk

sfdisk es un manipulador de la tabla de particiones del disco.

swapoff

swapoff desactiva los dispositivos y ficheros de paginación e intercambio.

swapon

swapon activa los dispositivos y ficheros de paginación e intercambio.

tunelp

tunelp establece varios parámetros para el dispositivo de puerto paralelo (LP).

ul

ul lee un fichero y traduce las ocurrencias de marcas de texto a la secuencia que indica subrayado para el terminal en uso.

umount

umount desmonta un sistema de ficheros o un directorio montado.

vidmode

vidmode muestra y establece el modo de vídeo.

whereis

whereis localiza el binario, la fuente y la página del manual de un comando.

write

write envía un mensaje a otro usuario, si ese usuario tiene activada la escritura (normalmente mediante mesg).

Dependencias de instalación de Util−linux

Última versión comprobada: 2.11n.

Bash: sh
Binutils: as, ld
Diffutils: cmp
Fileutils: chgrp, chmod, cp, install, ln, mv, rm
Gettext: msgfmt, xgettext
Gcc: cc, cc1, collect2, cpp, cpp0
Glibc: rpcgen
Grep: grep
Make: make
Sed: sed
Sh−utils: uname, whoami
Textutils: cat

Vim

Localización oficial para descarga

Vim (6.1):
ftp://ftp.vim.org/pub/editors/vim/unix/

Parche para Vim (6.1):
ftp://ftp.linuxfromscratch.org/lfs−packages/cvs/
http://downloads.linuxfromscratch.org/

Contenido de Vim

Última versión comprobada: 6.1.

El paquete Vim contiene un editor de texto configurable construido para obtener una eficiente edición del texto.

Vim instala lo siguiente:

Programas

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, tcltags, vi (enlace a vim), view (enlace a vim), vim, vim132, vim2html.pl, vimdiff (enlace a vim), vimm, vimspell.sh, vimtutor y xxd

Alternativas a Vim

- emacs, joe y nano

  http://www.escomposlinux.org/lfs−es/blfs−es−CVS/postlfs/editors.html (en castellano)

  http://beyond.linuxfromscratch.org/view/cvs/postlfs/editors.html (en inglés)

Contenido de Vim

Última versión comprobada: 6.1.

Descripción de los programas

efm_filter.pl

efm_filter.pl es un filtro que lee de la entrada estándar, copia a la salida estándar y crea un fichero de error que puede ser leido por vim.

efm_perl.pl

efm_perl.pl formatea los mensajes de error del intérprete Perl para usarlos con el modo "quickfix" de vim.

ex

ex arranca vim en modo Ex.

less.sh

less.sh es un guión que arranca vim con less.vim.

mve.awk

mve.awk procesa los errores de vim.

pltags.pl

pltags.pl crea un fichero de etiquetas para el código Perl, de modo que pueda usarse con vim.

ref

ref comprueba la ortografía de los argumentos que se le pasan.

rview

rview es una versión restringida de view. No pueden ejecutarse comandos del intérprete de comandos y vim no puede ser suspendido.

rvim

rvim es una versión restringida de vim. No pueden ejecutarse comandos del intérprete de comandos y vim no puede ser suspendido.

shtags.pl

shtags.pl genera un fichero de etiquetas para los guiones Perl.

tcltags

tcltags es un fichero de etiquetas para el código TCL.

vi

vi arranca vim en modo compatible con vi.

view

view arranca vim en modo de sólo lectura.

vim

vim arranca vim de la manera normal.

vim132

vim132 arranca vim con el terminal en modo de 132 columnas.

vim2html.pl

vim2html.pl convierte la documentación de vim a HTML.

vimdiff

vimdiff edita dos o tres versiones de un fichero con vim y muestra las diferencias.

vimm

vimm activa el modelo de entrada del buscador de DEC en un terminal remoto.

vimspell.sh

vimspell.sh es un guión que corrige un fichero y genera las sentencias de sintaxis necesarias para resaltar las palabras en vim.

vimtutor

vimtutor arranca el tutorial de vim.

xxd

xxd genera un volcado hexadecimal o hace lo contrario.

Dependencias de instalación de Vim

Última versión comprobada: 6.0.

Bash: sh
Binutils: as, ld, strip
Diffutils: cmp, diff
Fileutils: chmod, cp, ln, mkdir, mv, rm, touch
Find: find
Gcc: cc1, collect2, cpp0, gcc
Grep: egrep, grep
Make: make
Net−tools: hostname
Sed: sed
Sh−utils: echo, expr, uname, whoami
Textutils: cat, tr, wc

Zlib

Localización oficial para descarga

Zlib (1.1.4):
http://www.gzip.org/zlib/

Contenido de Zlib

Última versión comprobada: 1.1.4.

El paquete Zlib contiene la librería zlib, utilizada por muchos programas para realizar las funciones de compresión y descompresión..

Zlib instala lo siguiente:

Librerías

libz[a,so]

Contenido de Zlib

Última versión comprobada: 1.1.4.

Descripción de la librería

libz

Esta es la librería zlib, que muchos programas usan para sus funciones de compresión y descompresión.

Dependencias de instalación de Zlib

Dependencias no comprobadas todavía.