Linux From Scratch

Versión 12.3

Publicado el 5 de marzo de 2025

Creado por Gerard Beekmans

Editor jefe: Bruce Dubbs

Linux From Scratch: Versión 12.3: Publicado el 5 de marzo de 2025

Creado por Gerard Beekmans y Editor en Jefe: Bruce Dubbs

Prefacio

Prólogo

Mi camino para aprender y comprender mejor Linux comenzó en 1998. Acababa de instalar mi primera distribución de Linux y enseguida me intrigó el concepto y la filosofía que lo sustentaban.

Siempre hay muchas maneras de lograr una misma tarea. Lo mismo puede decirse de las distribuciones de Linux. A lo largo de los años han existido muchísimas. Algunas aún existen, otras se han transformado en algo diferente, y otras han quedado relegadas al olvido. Todas hacen las cosas de forma diferente para adaptarse a las necesidades de su público objetivo. Debido a la gran variedad de maneras de lograr el mismo objetivo, empecé a darme cuenta de que ya no tenía que limitarme a una sola implementación.

Antes de descubrir Linux, simplemente tolerábamos los problemas de otros sistemas operativos, ya que no había otra opción. Era lo que era, te gustara o no. Con Linux, empezó a surgir el concepto de la elección. Si algo no te gustaba, eras libre, incluso te animaban, a cambiarlo.

Probé varias distribuciones y no me decidía por ninguna. Eran sistemas excelentes por sí mismos. Ya no era una cuestión de lo correcto o lo incorrecto. Se había convertido en una cuestión de gustos personales. Con tantas opciones disponibles, se hizo evidente que no habría un solo sistema perfecto para mí. Así que me propuse crear mi propio sistema Linux que se ajustara por completo a mis preferencias.

Para convertirlo en mi propio sistema, decidí compilarlo todo desde el código fuente en lugar de usar paquetes binarios precompilados. Este sistema Linux "perfecto" tendría las fortalezas de varios sistemas sin sus aparentes debilidades. Al principio, la idea era bastante abrumadora. Seguí convencido de que un sistema así podía construirse.

Tras resolver problemas como las dependencias circulares y los errores de compilación, finalmente construí un sistema Linux a medida. Era completamente operativo y perfectamente utilizable como cualquier otro sistema Linux del momento. Pero era mi propia creación. Fue muy satisfactorio haber creado un sistema así yo mismo. Lo único mejor habría sido crear cada pieza de software yo mismo. Esta fue la mejor alternativa.

Al compartir mis objetivos y experiencias con otros miembros de la comunidad Linux, se hizo evidente un interés constante en estas ideas. Rápidamente se hizo evidente que estos sistemas Linux personalizados no solo satisfacen los requisitos específicos del usuario, sino que también constituyen una oportunidad de aprendizaje ideal para que programadores y administradores de sistemas mejoren sus conocimientos de Linux. De este interés creciente nació el Proyecto Linux Desde Cero.

Este libro, "Linux Desde Cero", es el núcleo central de dicho proyecto. Proporciona la información y las instrucciones necesarias para diseñar y construir tu propio sistema. Si bien este libro proporciona una plantilla que dará como resultado un sistema que funcione correctamente, tienes la libertad de modificar las instrucciones a tu gusto, lo cual es, en parte, una parte importante de este proyecto.

Tú tienes el control; nosotros solo te ayudamos a comenzar tu propio camino.

Espero sinceramente que lo pases genial trabajando en tu propio sistema Linux Desde Cero y que disfrutes de los numerosos beneficios de tener un sistema verdaderamente tuyo.

--
Gerard Beekmans
gerard@linuxfromscratch.org

Audiencia

Hay muchas razones por las que querrías leer este libro. Una de las preguntas que muchos se hacen es: "¿Por qué complicarse la vida construyendo un sistema Linux desde cero cuando puedes simplemente descargar e instalar uno ya existente?"

Una razón importante de la existencia de este proyecto es ayudarte a aprender cómo funciona un sistema Linux desde dentro.

Construir un sistema LFS ayuda a demostrar qué hace que Linux funcione y cómo todo funciona en conjunto y depende entre sí. Una de las mejores cosas que esta experiencia de aprendizaje puede ofrecerte es la posibilidad de personalizar un sistema Linux para que se ajuste a tus necesidades.

Otro beneficio clave de LFS es que te da el control del sistema sin depender de la implementación de Linux de otro. Con LFS, tienes el control total. Tú decides cada aspecto de tu sistema.

LFS te permite crear sistemas Linux muy compactos. Con otras distribuciones, a menudo te ves obligado a instalar una gran cantidad de programas que no usas ni entiendes. Estos programas desperdician recursos. Podrías argumentar que con los discos duros y CPU actuales, el desperdicio de recursos ya no es un problema. Sin embargo, a veces, el tamaño del sistema sigue siendo una limitación, como mínimo. Piensa en CD de arranque, memorias USB y sistemas integrados. En estas áreas, LFS puede ser beneficioso. Otra ventaja de un sistema Linux a medida es la seguridad. Al compilar todo el sistema desde el código fuente, puede auditarlo todo y aplicar todos los parches de seguridad que desee. No tiene que esperar a que alguien compile paquetes binarios que solucionen una vulnerabilidad de seguridad. A menos que examine el parche y lo implemente usted mismo, no tiene garantía de que el nuevo paquete binario se haya compilado correctamente y solucione el problema adecuadamente.

El objetivo de Linux From Scratch es construir un sistema básico completo y usable. Si no desea construir su propio sistema Linux desde cero, puede beneficiarse de la información de este libro.

Hay tantas buenas razones para construir su propio sistema LFS que es imposible enumerarlas todas aquí. En definitiva, la formación es, con mucho, la razón más importante. A medida que avance en su experiencia con LFS, descubrirá el poder que la información y el conocimiento pueden aportar.

Arquitecturas de destino de LFS

Las principales arquitecturas de destino de LFS son las CPU AMD/Intel x86 (32 bits) y x86_64 (64 bits). Por otro lado, se sabe que las instrucciones de este libro también funcionan, con algunas modificaciones, con las CPU Power PC y ARM.

Para construir un sistema que utilice una de estas CPU alternativas, el requisito principal, además de los que se indican en la página siguiente, es un sistema Linux existente, como una instalación anterior de LFS, Ubuntu, Red Hat/Fedora, SuSE o alguna otra distribución que se adapte a esa arquitectura. (Tenga en cuenta que una distribución de 32 bits puede instalarse y utilizarse como sistema host en un ordenador AMD/Intel de 64 bits).

La ventaja de construir en un sistema de 64 bits, en comparación con uno de 32 bits, es mínima. Por ejemplo, en una compilación de prueba de LFS-9.1 en un sistema con CPU Core i7-4790 y 4 núcleos, se midieron las siguientes estadísticas:

Arquitectura	Tiempo de compilación	Tamaño de compilación
32 bits	239,9 minutos	3,6 GB
64 bits	233,2 minutos	4,4 GB

Como puede observar, en el mismo hardware, la compilación de 64 bits es solo un 3 % más rápida (y un 22 % más grande) que la de 32 bits. Si planea usar LFS como servidor LAMP o cortafuegos, una CPU de 32 bits puede ser suficiente. Por otro lado, varios paquetes de BLFS ahora requieren más de 4 GB de RAM para compilarse o ejecutarse; si planea usar LFS como escritorio, los autores de LFS recomiendan compilar un sistema de 64 bits.

La compilación predeterminada de 64 bits resultante de LFS es un sistema de 64 bits puro. Es decir, solo admite ejecutables de 64 bits. Construir un sistema multibiblioteca requiere compilar varias aplicaciones dos veces: una para un sistema de 32 bits y otra para uno de 64 bits. Esto no se admite directamente en LFS porque interferiría con el objetivo educativo de proporcionar las instrucciones mínimas necesarias para un sistema Linux básico. Algunos editores de LFS/BLFS mantienen una bifurcación multibiblioteca de LFS, accesible en Su *Version r12.3-943-multilib*. Pero ese es un tema avanzado (https://www.linuxfromscratch.org/~thomas/multilib/index.html).

Prerrequisitos

Construir un sistema LFS no es tarea sencilla. Requiere cierto nivel de conocimientos previos de administración de sistemas Unix para resolver problemas y ejecutar correctamente los comandos indicados. En particular, como mínimo, debería saber usar la línea de comandos (shell) para copiar o mover archivos y directorios, listar el contenido de directorios y archivos, y cambiar el directorio actual. También se espera que sepa usar e instalar software de Linux.

Dado que el libro de LFS presupone al menos este nivel básico de conocimientos, es poco probable que los diversos foros de soporte de LFS le proporcionen mucha ayuda en estas áreas. Es probable que sus preguntas sobre estos conocimientos básicos queden sin respuesta (o simplemente se le remitirá a la lista de lecturas previas esenciales de LFS).

Antes de crear un sistema LFS, le recomendamos leer estos artículos:

- Manual de desarrollo de software:
  https://tldp.org/HOWTO/Software-Building-HOWTO.html
  Esta es una guía completa para crear e instalar paquetes de software genéricos de Unix en Linux. Aunque se escribió hace tiempo, ofrece un buen resumen de las técnicas básicas para crear e instalar software.

- Guía para principiantes sobre la instalación desde el código fuente:
  https://moi.vonos.net/linux/beginners-installing-from-source/
  Esta guía proporciona un buen resumen de las habilidades y técnicas básicas necesarias para crear software a partir del código fuente.

LFS y estándares

La estructura de LFS sigue los estándares de Linux al máximo. Los principales estándares son:

- POSIX.1-2008.
- Estándar de Jerarquía del Sistema de Archivos (FHS) Versión 3.0
- Base Estándar de Linux (LSB) Versión 5.0 (2015)

El LSB consta de cuatro especificaciones independientes: Núcleo, Escritorio, Idiomas e Imágenes. Algunas partes de las especificaciones del Núcleo y del Escritorio son específicas de cada arquitectura. También existen dos especificaciones de prueba: GTK3 y Gráficos.

El LFS intenta cumplir con las especificaciones del LSB para las arquitecturas IA32 (x86 de 32 bits) o AMD64 (x86_64) descritas en la sección anterior.

Nota

Mucha gente no está de acuerdo con estos requisitos. El objetivo principal del LSB es garantizar que el software propietario pueda instalarse y ejecutarse en un sistema compatible. Dado que el LFS se basa en el código fuente, el usuario tiene control total sobre los paquetes que desea; puede optar por no instalar algunos paquetes especificados por el LSB.

Si bien es posible crear un sistema completo que supere las pruebas de certificación LSB desde cero, esto no es posible sin muchos paquetes adicionales que exceden el alcance del manual de LFS. Las instrucciones de instalación de algunos de estos paquetes adicionales se pueden encontrar en BLFS.

Paquetes proporcionados por LFS necesarios para cumplir con los requisitos de LSB:

LSB Core:	*Bash, Bc, Binutils, Coreutils, Diffutils, File, Findutils, Gawk, GCC, Gettext, Glibc, Grep, Gzip, M4, Man-DB, Procps, Psmisc, Sed, Shadow, SysVinit, Tar, Util-linux, Zlib*
LSB Desktop:	*Ninguno*
LSB Languages:	*Perl*
LSB Imaging:	*Ninguno*

LSB Gtk3 y LSB Graphics (uso de prueba): Ninguno

Paquetes proporcionados por BLFS necesarios para cumplir con los requisitos de LSB

LSB Core: *At, Batch (parte de At), archivos de inicio de BLFS Bash, Cpio, Ed, Fcrontab, LSB-Tools, NSPR, NSS, Linux-PAM, Pax, Sendmail (o Postfix o Exim), Time.*

LSB Desktop: *Alsa, ATK, Cairo, Desktop-file-utils, Freetype, Fontconfig, Gdk-pixbuf, Glib2, GLU, Icon-naming-utils, Libjpeg-t urbo, Libxml2, Mesa, Pango, Xdg-utils, Xorg.*

LSB Languages: *Libxml2, Libxslt.*

LSB Imaging: *CUPS, Cups-filters, Ghostscript, SANE.*

LSB Gtk3 y LSB Graphics (uso de prueba): GTK+3.

Componentes no proporcionados o proporcionados opcionalmente por LFS o BLFS necesarios para cumplir con los requisitos Requisitos de LSB

LSB Core: *install_initd, libcrypt.so.1 (puede proporcionarse con instrucciones opcionales para el paquete LFS Libxcrypt), libncurses.so.5 (puede proporcionarse con instrucciones opcionales para el paquete LFS Ncurses), libncursesw.so.5 (pero libncursesw.so.6 lo proporciona el paquete LFS Ncurses).*

LSB Desktop: *libgdk-x11-2.0.so (pero libgdk-3.so lo proporciona el paquete BLFS GTK+-3), libgtk-x11-2.0.so (pero libgtk-3.so y libgtk-4.so lo proporcionan los paquetes BLFS GTK+-3 y GTK-4), libpng12.so (pero libpng16.so lo proporciona el paquete BLFS Libpng). libQt*.so.4 (pero l ibQt6*.so.6 lo proporciona el paquete Qt6 de BLFS), libtiff.so.4 (pero libtiff.so.6 lo proporciona el paquete Libtiff de BLFS).*

Lenguajes LSB: */usr/bin/python (LSB requiere Python 2, pero LFS y BLFS solo ofrecen Python 3).*

Imágenes LSB: *Ninguna.*

Gráficos LSB GTK3 y LSB (uso de prueba): libpng15.so (pero libpng16.so lo proporciona el paquete Libpng de BLFS).

Justificación de los paquetes del libro

El objetivo de LFS es construir un sistema básico completo y usable, que incluya todos los paquetes necesarios para replicarse, y proporcionar una base relativamente mínima a partir de la cual personalizar un sistema más completo según las preferencias del usuario. Esto no significa que LFS sea el sistema más pequeño posible. Se incluyen varios paquetes importantes que, en sentido estricto, no son obligatorios. La siguiente lista documenta las razones por las que se ha incluido cada paquete en el libro.

- Acl.
  Este paquete contiene utilidades para administrar Listas de Control de Acceso, que se utilizan para definir permisos de acceso discrecionales de granularidad precisa para archivos y directorios.

- Attr.
  Este paquete contiene programas para administrar atributos extendidos en objetos del sistema de archivos.

- Autoconf.
  Este paquete proporciona programas para generar scripts de shell que pueden configurar automáticamente el código fuente a partir de una plantilla de desarrollador. A menudo es necesario reconstruir un paquete después de actualizar el procedimiento de compilación.

- Automake.
  Este paquete contiene programas para generar archivos Make a partir de una plantilla. A menudo es necesario reconstruir un paquete después de actualizar el procedimiento de compilación.

- Bash.
  Este paquete satisface un requisito básico de LSB para proporcionar una interfaz Bourne Shell al sistema. Se eligió sobre otros paquetes de shell debido a su uso común y amplias capacidades.

- Bc.
  Este paquete proporciona un lenguaje de procesamiento numérico de precisión arbitraria. Satisface un requisito para compilar el kernel de Linux.

- Binutils.
  Este paquete proporciona un enlazador, un ensamblador y otras herramientas para gestionar archivos objeto. Los programas de este paquete son necesarios para compilar la mayoría de los paquetes en un sistema LFS.

- Bison.
  Este paquete contiene la versión GNU de yacc (Yet Another Compiler Compiler), necesaria para compilar varios programas LFS.

- Bzip2.
  Este paquete contiene programas para comprimir y descomprimir archivos. Es necesario para descomprimir muchos paquetes LFS.

- Check.
  Este paquete proporciona una herramienta de prueba para otros programas.
- Coreutils.
  Este paquete contiene varios programas esenciales para visualizar y manipular archivos y directorios. Estos programas son necesarios para la gestión de archivos desde la línea de comandos y para los procedimientos de instalación de todos los paquetes en LFS.
- DejaGNU.
  Este paquete proporciona un marco para probar otros programas.
- Diffutils.
  Este paquete contiene programas que muestran las diferencias entre archivos o directorios. Estos programas se pueden usar para crear parches y también se utilizan en los procedimientos de compilación de muchos paquetes.
- E2fsprogs.
  Este paquete proporciona utilidades para gestionar los sistemas de archivos ext2, ext3 y ext4. Estos son los sistemas de archivos más comunes y ampliamente probados que Linux soporta.
- Expat.
  Este paquete proporciona una biblioteca de análisis de XML relativamente pequeña. Es necesaria para el módulo XML::Parser de Perl.
- Expect.
  Este paquete contiene un programa para ejecutar diálogos programados con otros programas interactivos. Se utiliza comúnmente para probar otros paquetes.
- File.
  Este paquete contiene una utilidad para determinar el tipo de uno o más archivos. Algunos paquetes la necesitan en sus scripts de compilación.
- Findutils.
  Este paquete proporciona programas para buscar archivos en un sistema de archivos. Se utiliza en los scripts de compilación de muchos paquetes.
- Flex.
  Este paquete contiene una utilidad para generar programas que reconocen patrones en texto. Es l a versión GNU del programa lex (analizador léxico). Es necesario para compilar varios paquetes LFS.
- Gawk.
  Este paquete proporciona programas para manipular archivos de texto. Es la versión GNU de awk (Aho-Weinberg-Kernighan). Se utiliza en los scripts de compilación de muchos otros paquetes.
- GCC.
  Esta es la Colección de Compiladores GNU. Contiene los compiladores de C y C++, así como varios otros no compilados por LFS.

- GDBM.
  Este paquete contiene la biblioteca GNU Database Manager. La utiliza otro paquete de LFS: Man-DB.
- Gettext.
  Este paquete proporciona utilidades y bibliotecas para la internacionalización y localización de numerosos paquetes.
- Glibc.
  Este paquete contiene la biblioteca principal de C. Los programas de Linux no se ejecutarán sin ella.
- GMP.
  Este paquete proporciona bibliotecas matemáticas que ofrecen funciones útiles para cálculos aritméticos de precisión arbitraria. Es necesario para compilar GCC.
- Gperf.
  Este paquete produce un programa que genera una función hash perfecta a partir de un conjunto de claves. Es requerido por Udev.
- Grep.
  Este paquete contiene programas para buscar archivos. Estos programas se utilizan en la mayoría de los scripts de compilación de los paquetes.
- Groff.
  Este paquete proporciona programas para procesar y formatear texto. Una función importante de estos programas es formatear páginas de manual.
- GRUB.
  Este es el Gran Cargador de Arranque Unificado. Es el más flexible de los cargadores de arranque disponibles.
- Gzip.
  Este paquete contiene programas para comprimir y descomprimir archivos. Es necesario para descomprimir muchos paquetes en LFS.
- Iana-etc.
  Este paquete proporciona datos para servicios y protocolos de red. Es necesario para habilitar capacidades de red adecuadas.
- Inetutils.
  Este paquete proporciona programas para la administración básica de red.
- Intltool.
  Este paquete proporciona herramientas para extraer cadenas traducibles de archivos fuente.
- IProute2.
  Este paquete contiene programas para redes IPv4 e IPv6 básicas y avanzadas. Fue elegido en lugar del otro paquete de herramientas de red común (net-tools) por sus capacidades IPv6.

- Kbd.
  Este paquete produce archivos de tablas de teclas, utilidades de teclado para teclados no estadounidenses y diversas fuentes de consola.

- Kmod.
  Este paquete proporciona los programas necesarios para administrar los módulos del kernel de Linux.

- Less.
  Este paquete contiene un visor de archivos de texto muy útil que permite desplazarse hacia arriba o hacia abajo al visualizar un archivo. Muchos paquetes lo utilizan para paginar la salida.

- Libcap.
  Este paquete implementa las interfaces de espacio de usuario para las capacidades POSIX 1003.1e disponibles en los kernels de Linux.

- Libelf.
  El proyecto elfutils proporciona bibliotecas y herramientas para archivos ELF y datos DWARF. La mayoría de las utilidades de este paquete están disponibles en otros paquetes, pero la biblioteca es necesaria para compilar el kernel de Linux con la configuración predeterminada (la más eficiente).

- Libffi.
  Este paquete implementa una interfaz de programación portátil de alto nivel para diversas convenciones de llamada. Algunos programas pueden no saber, al compilar, qué argumentos se deben pasar a una función. Por ejemplo, a un intérprete se le puede indicar en tiempo de ejecución el número y los tipos de argumentos utilizados para llamar a una función determinada. Libffi puede utilizarse en estos programas para conectar el programa intérprete con el código compilado.

- Libpipeline.
  El paquete Libpipeline proporciona una biblioteca para manipular las canalizaciones de subprocesos de forma flexible y cómoda. Es necesario para el paquete Man-DB.

- Libtool.
  Este paquete contiene el script de soporte para bibliotecas genéricas de GNU. Consiste en la complejidad del uso de bibliotecas compartidas en una interfaz consistente y portátil. Es necesario para las suites de pruebas de otros paquetes LFS.

- Libxcrypt.
  Este paquete proporciona la biblioteca libcrypt, necesaria para varios paquetes (en particular, Shadow) para el hash de contraseñas.

  Reemplaza la implementación obsoleta de libcrypt en Glibc.

- Kernel Linux.
  Este paquete representa el sistema operativo. Representa Linux en el entorno GNU/Linux.

- M4.
  Este paquete proporciona un procesador de macros de texto general, útil como herramienta de compilación para otros programas.
- Make.
  Este paquete contiene un programa para dirigir la compilación de paquetes. Es necesario para casi todos los paquetes de LFS.
- Man-DB.
  Este paquete contiene programas para buscar y visualizar páginas de manual. Se eligió en lugar del paquete man debido a sus superiores capacidades de internacionalización. Proporciona el programa man.
- Man-Pages.
  Este paquete proporciona el contenido de las páginas de manual básicas de Linux.
- Meson.
  Este paquete proporciona una herramienta de software para automatizar la compilación de software. El objetivo principal de Meson es minimizar el tiempo que los desarrolladores de software dedican a configurar un sistema de compilación. Es necesario para compilar Systemd, así como muchos paquetes BLFS.
- MPC.
  Este paquete proporciona funciones aritméticas para números complejos. Es requerido por GCC.
- MPFR.
  Este paquete contiene funciones para aritmética de precisión múltiple. Es requerido por GCC.
- Ninja.
  Este paquete proporciona un sistema de compilación pequeño enfocado en la velocidad. Está diseñado para que sus archivos de entrada sean generados por un sistema de compilación de alto nivel y para ejecutar las compilaciones lo más rápido posible. Este paquete es requerido por Meson.
- Ncurses.
  Este paquete contiene bibliotecas para la gestión de pantallas de caracteres independiente de la terminal. Se utiliza a menudo para proporcionar control del cursor en un sistema de menús. Es necesario para varios paquetes de LFS.
- Openssl.
  Este paquete proporciona herramientas de gestión y bibliotecas relacionadas con la criptografía. Estas proporcionan funciones criptográficas a otros paquetes, incluido el kernel de Linux.
- Patch.
  Este paquete contiene un programa para modificar o crear archivos mediante la aplicación de un archivo de parche, normalmente creado por el programa diff. Es necesario para el proceso de compilación de varios paquetes de LFS.

- Perl.
  Este paquete es un intérprete para el lenguaje de ejecución PERL. Es necesario para la instalación y los conjuntos de pruebas de varios paquetes de LFS.
- Pkgconf.
  Este paquete contiene un programa que ayuda a configurar los indicadores del compilador y del enlazador para las bibliotecas de desarrollo.

  El programa puede utilizarse como sustituto directo de pkg-config, necesario para el sistema de compilación de muchos paquetes. Su mantenimiento es más activo y ligeramente más rápido que el del paquete Pkg-config original.
- Procps-NG.
  Este paquete contiene programas para la monitorización de procesos. Estos programas son útiles para la administración del sistema y también los utilizan los scripts de arranque de LFS.
- Psmisc.
  Este paquete produce programas para mostrar información sobre los procesos en ejecución. Estos programas son útiles para la administración del sistema.
- Python 3.
  Este paquete proporciona un lenguaje interpretado con una filosofía de diseño que prioriza la legibilidad del código.
- Readline.
  Este paquete es un conjunto de bibliotecas que ofrecen funciones de edición e historial desde la línea de comandos. Es utilizado por Bash.
- Sed.
  Este paquete permite editar texto sin abrirlo en un editor de texto. También es necesario para los scripts de configuración de muchos paquetes LFS.
- Shadow.
  Este paquete contiene programas para gestionar contraseñas de forma segura.
- Sysklogd.
  Este paquete proporciona programas para registrar mensajes del sistema, como los emitidos por el núcleo o los procesos demonio cuando se producen eventos inusuales.
- SysVinit.
  Este paquete proporciona el programa init, el padre de todos los demás procesos en un sistema Linux en ejecución.
- Udev.
  Este paquete es un administrador de dispositivos. Controla dinámicamente la propiedad, los permisos, los nombres y los enlaces simbólicos de los nodos de dispositivo en el directorio /dev cuando se añaden o eliminan dispositivos del sistema.

- Tar.
  Este paquete proporciona funciones de archivado y extracción de prácticamente todos los paquetes utilizados en LFS.
- Tcl.
  Este paquete contiene el lenguaje de comandos de herramientas (Tool Command Language) utilizado en numerosas suites de pruebas.
- Texinfo.
  Este paquete proporciona programas para leer, escribir y convertir páginas de información. Se utiliza en los procedimientos de instalación de numerosos paquetes LFS.
- Util-linux.
  Este paquete contiene diversas utilidades, entre ellas, utilidades para gestionar sistemas de archivos, consolas, particiones y mensajes.
- Vim.
  Este paquete proporciona un editor. Fue elegido por su compatibilidad con el editor clásico vi y su gran cantidad de potentes funciones. El editor es una elección muy personal para muchos usuarios. Se puede sustituir por cualquier otro editor si se desea.
- Wheel.
  Este paquete proporciona un módulo de Python que es la implementación de referencia del estándar de empaquetado wheel de Python.
- XML::Parser.
  Este paquete es un módulo de Perl que interactúa con Expat.
- XZ Utils.
  Este paquete contiene programas para comprimir y descomprimir archivos. Proporciona la mayor compresión disponible y es útil para descomprimir paquetes en formato XZ o LZMA.
- Zlib.
  Este paquete contiene rutinas de compresión y descompresión utilizadas por algunos programas.
- Zstd.
  Este paquete proporciona rutinas de compresión y descompresión utilizadas por algunos programas. Ofrece altas tasas de compresión y una amplia gama de compensaciones entre compresión y velocidad.

Tipografía

Para facilitar la comprensión, se utilizan algunas convenciones tipográficas a lo largo de este libro. Esta sección contiene algunos ejemplos del formato tipográfico presente en Linux From Scratch.

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

Este formato de texto está diseñado para escribirse exactamente como se ve, a menos que se indique lo contrario en el texto circundante. También se utiliza en las secciones de explicación para identificar a qué comando se hace referencia.

En algunos casos, una línea de comandos se extiende a dos o más líneas físicas gracias a una barra invertida al final de la línea.

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

Tenga en cuenta que la barra invertida debe ir seguida inmediatamente de un **ENTER** y nada más. Otros caracteres, como los espacios en blanco o los tabuladores, generarán resultados incorrectos.

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

Este formato de texto (texto de ancho fijo) muestra la salida en pantalla, generalmente como resultado de los comandos ejecutados. Este formato también se utiliza para mostrar nombres de archivos, como /etc/ld.so.conf.

Nota

Configure su navegador para mostrar texto de ancho fijo con una buena fuente monoespaciada "font-size="9ptd", con la que podrá distinguir claramente los glifos de *Il1* u *O0*.

Énfasis

Este formato de texto se utiliza para varios propósitos en el libro. Su propósito principal es enfatizar puntos o elementos importantes.

https://www.linuxfromscratch.org/

Este formato se utiliza para hipervínculos tanto dentro de la comunidad LFS como a páginas externas. Incluye tutoriales, ubicaciones de descarga y sitios web.

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

Este formato se utiliza al crear archivos de configuración. El primer comando indica al sistema que cree el archivo *$LFS/etc/group* a partir de lo que se escriba en las líneas siguientes hasta que se encuentre la secuencia de fin de archivo (EOF). Por lo tanto, toda esta sección se escribe generalmente como se ve.

<TEXTO REEMPLAZADO>

Este formato se utiliza para encapsular texto que no se debe escribir como se ve o para operaciones de copiar y pegar.

[*TEXTO OPCIONAL*]

Este formato se utiliza para encapsular texto opcional.

passwd(5)

Este formato se utiliza para hacer referencia a una página específica del manual (man). El número entre paréntesis indica una sección específica de los manuales. Por ejemplo, **passwd** tiene dos páginas de manual. Según las instrucciones de instalación de LFS, estas dos páginas de manual se encuentran en */usr/share/man/man1/passwd.1 y /usr/share/man/man5/passwd.5*. Cuando el libro usa *passwd(5)*, se refiere específicamente a */usr/share/man/man5/passwd.5*. **man passwd** imprimirá la primera página de manual que encuentre que coincida con "passwd", que será */usr/share/man/man1/passwd.1*. Para este ejemplo, deberá ejecutar **man 5 passwd** para leer la página especificada. Tenga en cuenta que la mayoría de las páginas de manual no tienen nombres de página duplicados en diferentes secciones. Por lo tanto, **man <nombre del programa>** suele ser suficiente. En el libro LFS, estas referencias a páginas de manual también son hipervínculos, por lo que al hacer clic en dicha referencia se abrirá la página de manual representada en HTML desde las páginas del manual de Arch Linux.

Estructura

Este libro se divide en las siguientes partes:

Parte I - Introducción

La Parte I explica algunas notas importantes sobre cómo proceder con la instalación de LFS. Esta sección también proporciona metainformación sobre el libro.

Parte II - Preparación para la compilación

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

Parte III - Compilación de la cadena de herramientas cruzadas de LFS y las herramientas temporales

La Parte III proporciona instrucciones para compilar las herramientas necesarias para construir el sistema LFS final.

Parte IV - Compilación del sistema LFS

La Parte IV guía al lector a través de la compilación del sistema LFS: compilar e instalar todos los paquetes uno por uno, configurar los scripts de arranque e instalar el kernel. El sistema Linux resultante es la base sobre la que se puede compilar otro software para expandir el sistema según se desee. Al final de este libro, se incluye una referencia fácil de usar que enumera todos los programas, bibliotecas y archivos importantes instalado.

Parte V - Apéndices

La Parte V proporciona información sobre el libro, incluyendo acrónimos y términos, agradecimientos, dependencias de los paquetes, una lista de scripts de arranque de LFS, licencias para la distribución del libro y un índice completo de paquetes, programas, bibliotecas y scripts.

Erratas y avisos de seguridad

El software utilizado para crear un sistema LFS se actualiza y mejora constantemente. Las advertencias de seguridad y las correcciones de errores podrían estar disponibles después de la publicación del libro de LFS. Para comprobar si las versiones de los paquetes o las instrucciones de esta versión de LFS necesitan alguna modificación (para reparar vulnerabilidades de seguridad o corregir otros errores), visite *https://www.linuxfromscratch.org/lfs/errata/12.3/* antes de continuar con la compilación. Debe tener en cuenta los cambios mostrados y aplicarlos a las secciones pertinentes del libro a medida que compila el sistema LFS.

Además, los editores de Linux From Scratch mantienen una lista de vulnerabilidades de seguridad descubiertas después de la publicación de un libro. Para consultar la lista, visite *https://www.linuxfromscratch.org/lfs/advisories/* antes de continuar con la compilación. Debe aplicar los cambios sugeridos por los avisos a las secciones relevantes del libro a medida que compila el sistema LFS. Si va a utilizar el sistema LFS como un sistema de escritorio o servidor real, debe seguir consultando los avisos y corregir cualquier vulnerabilidad de seguridad, incluso cuando el sistema LFS esté completamente compilado.

Parte I. Introducción

Capítulo 1. Introducción

1.1. Cómo construir un sistema LFS

El sistema LFS se construirá utilizando una distribución de Linux ya instalada (como Debian, OpenMandriva, Fedora u openSUSE). Este sistema Linux existente (el host) se utilizará como punto de partida para proporcionar los programas necesarios, incluyendo un compilador, un enlazador y un intérprete de comandos, para construir el nuevo sistema. Seleccione la opción "desarrollo" durante la instalación de la distribución para incluir estas herramientas.

Nota

Existen muchas maneras de instalar una distribución de Linux y las opciones predeterminadas no suelen ser las óptimas para construir un sistema LFS. Para obtener sugerencias sobre cómo configurar una distribución comercial, consulte: *https://www.linuxfromscratch.org/hints/downloads/files/partitioning-for-lfs.txt*.

Como alternativa a instalar una distribución independiente en su equipo, puede utilizar un LiveCD de una distribución comercial.

El capítulo 2, Preparación del Sistema Host, de este libro describe cómo crear una nueva partición y un sistema de archivos nativos de Linux, donde se compilará e instalará el nuevo sistema LFS.

El capítulo 3, Paquetes y parches,explica qué paquetes y parches deben descargarse para compilar un sistema LFS y cómo almacenarlos en el nuevo sistema de archivos.

El capítulo 4, Preparativos Finales, describe la configuración de un entorno de trabajo adecuado. Lea atentamente el capítulo 4, ya que explica varios aspectos importantes que debe tener en cuenta antes de comenzar a trabajar con el capítulo 5 y posteriores.

El capítulo 5, Compilación de una cadena de herramientas cruzadas, explica la instalación de la cadena de herramientas inicial (binutils, gcc y glibc) mediante técnicas de compilación cruzada para aislar las nuevas herramientas del sistema host.

El capítulo 6, Herramientas temporales para compilación cruzada, muestra cómo realizar la compilación cruzada de utilidades básicas utilizando la cadena de herramientas cruzada recién compilada.

El capítulo 7, Entrada al entorno chroot y creación de herramientas temporales adicionales, se centra en un entorno "chroot", donde utilizamos las nuevas herramientas para compilar el resto de las herramientas necesarias para crear el sistema LFS.

Este esfuerzo por aislar el nuevo sistema de la distribución host puede parecer excesivo. En las Notas técnicas de la cadena de compilación o Toolchain se ofrece una explicación técnica completa de por qué se hace esto.

El Capítulo 8, Instalación del software básico del sistema, se dedica a construir el sistema LFS completo. se dedica a construir el sistema LFS completo. Otra ventaja del entorno chroot es que

permite seguir usando el sistema host mientras se construye LFS. Mientras espera a que se completen las compilaciones de paquetes, puede seguir usando su ordenador con normalidad.

El Capítulo 9, Instalación del software básico del sistema; Configuración del sistema, ya para comenzar a finalizar la instalación, trata la configuración básica del sistema.

El Capítulo 10, Haciendo que el sistema LFS sea arrancable crea el kernel y configura el gestor de arranque; Crea el kernel y configura el gestor de arranque.

El Capítulo 11, Fin; Contiene información sobre cómo continuar con la experiencia LFS más allá de este libro. Una vez implementados los pasos de este capítulo, el ordenador está listo para arrancar en el nuevo sistema LFS.

Este fue el proceso en resumen. En los siguientes capítulos se presenta información detallada sobre cada paso. Los aspectos que ahora parecen complicados se aclararán y todo encajará a medida que comience su aventura con LFS.

1.2. Novedades desde la última versión

Aquí tienes una lista de los paquetes actualizados desde la última versión de LFS. **Actualizado a**:

• Bash-5.2.37.

• Bc-7.0.3.

• Binutils-2.44.

• Coreutils-9.6.

• Diffutils-3.11.

• E2fsprogs-1.47.2.

• Expat-2.6.4.

• File-5.46.

• Flit-core-3.11.0.

• Gawk-5.3.1.

• Gettext-0.24.

• Glibc-2.41.

• Iana-Etc-20250123.

• Inetutils-2.6.

• IPRoute2-6.13.0.

• Jinja2-3.1.5.

• Kbd-2.7.1.

• Kmod-34.

• Less-668.

• Libcap-2.73.

• Libelf from Elfutils-0.192.

• Libffi-3.4.7.

• Libpipeline-1.5.8.

• Libtool-2.5.4.

• Libxcrypt-4.4.38.

• Linux-6.13.4.

• Man-DB-2.13.0.

• Man-pages-6.12.

• MarkupSafe-3.0.2.

• Meson-1.7.0.

• OpenSSL-3.4.1.

• Perl-5.40.1.

• Procps-ng-4.0.5.

• Python-3.13.2.

• Setuptools-75.8.1.

• Shadow-4.17.3.

• Sysklogd-2.7.0.

• Systemd-257.3.

• SysVinit-3.14.

• Tcl-8.6.16.

• Texinfo-7.2.

• Tzdata-2025a.

• Udev from Systemd-257.3.

• Util-linux-2.40.4.

• Vim-9.1.1166.

• Wheel-0.45.1.

• Xz-5.6.4.

• Zstd-1.5.7.

Añadido:

•

Eliminado:

•

1.3. Registro de cambios

Esta es la versión 12.3 del libro "Linux From Scratch", del 5 de marzo de 2025. Si este libro tiene más de seis meses de antigüedad, probablemente ya esté disponible una versión más reciente y mejorada. Para obtener más información, consulte uno de los servidores de réplica en *https://www.linuxfromscratch.org/mirrors.html*.

A continuación, se muestra una lista de los cambios realizados desde la versión anterior del libro.

Entradas del registro de cambios:

• 05/03/2025.

    • [bdubbs] - Publicación de LFS-12.3.

• 02/03/2025.

    • [bdubbs] - Actualización a vim-9.1.1166 (Actualización de seguridad). Corrige el problema # 5666.

• 27/02/2025.

    • [bdubbs] - Actualización a zstd-1.5.7. Corrige el problema # 5652.

    • [bdubbs] - Actualización a systemd-257.3. Corrige el error # 5612.

    • [bdubbs] - Actualización a shadow-4.17.3. Corrige el error # 5660.

    • [bdubbs] - Actualización a setuptools-75.8.1. Corrige el error # 5662.

    • [bdubbs] - Actualización a linux-6.13.4. Corrige el error # 5647.

    • [bdubbs] - Actualización a kmod-34. Corrige el error # 5657.

    • [bdubbs] - Actualización a inetutils-2.6. Corrige el error # 5656.

    • [bdubbs] - Actualización a gettext-0.24. Corrige el error # 5661.

    • [bdubbs] - Actualización a flit_core-3.11.0. Corrige el error # 5654.

• 24/02/2025.

    • [bdubbs] - Actualización a man-pages-6.12. Corrige el problema # 5658.

• 19/02/2025.

    • [xry111] - Actualización a vim-9.1.1122 (Actualización de seguridad). Corrige el problema # 4500.

    • [xry111] - Actualización a man-pages-6.11. Corrige el problema # 5646.

• 13/02/2025.

    • [bdubbs] - Actualización a vim-9.1.1106. Corrige el problema # 4500.

    • [bdubbs] - Actualización a diffutils-3.11. Corrige el problema # 5639.

    • [bdubbs] - Actualización a libffi-3.4.7. Corrige el problema # 5642.

    • [bdubbs] - Actualización a Linux-6.13.2. Corrige el problema # 5643.

    - [bdubbs] - Actualización a Python3-3.13.2. Corrige el problema # 5640.
    - [bdubbs] - Actualización a Sysvinit-3.14. Corrige el problema # 5641.
- 02/02/2025.
    - [bdubbs] - Actualización a Vim-9.1.1071. Corrige el problema # 4500.
    - [bdubbs] - Actualización a iana-etc-20250123. Corrige el problema # 5006.
    - [bdubbs] - Actualización a Binutils-2.44.0. Corrige el problema # 5634.
    - [bdubbs] - Actualización a Coreutils-9.6. Corrige el problema # 5628.
    - [bdubbs] - Actualización a e2fsprogs-1.47.2. Corrige el problema # 5637.
    - [bdubbs] - Actualización a glibc-2.41. Corrige el problema # 5638.
    - [bdubbs] - Actualización a iproute2-6.13.0. Corrige el problema # 5631.
    - [bdubbs] - Actualización a libxcrypt-4.4.38. Corrige el problema # 5626.
    - [bdubbs] - Actualización a linux-6.13.1. Corrige el problema # 5629.
    - [bdubbs] - Actualización a man-pages-6.10. Corrige el problema # 5632.
    - [bdubbs] - Actualización a meson-1.7.0. Corrige el problema # 5636.
    - [bdubbs] - Actualización a perl-5.40.1. Corrige el problema # 5630.
    - [bdubbs] - Actualización a tcl8.6.16. Corrige el problema # 5635.
    - [bdubbs] - Actualización a tzdata2025a. Corrige el problema # 5627.
    - [bdubbs] - Actualización a xz-5.6.4. Corrige el problema # 5633.
- 15/01/2025.
    - [bdubbs] - Actualización a vim-9.1.1016. Corrige el problema # 4500.
    - [bdubbs] - Actualización a iana-etc-20250108. Corrige el problema # 5006.
    - [bdubbs] - Actualización a util-linux-2.40.4. Corrige el problema # 5624.
    - [bdubbs] - Actualización a sysvinit-3.13. Corrige el problema # 5621.
    - [bdubbs] - Actualización a sysklogd-2.7.0. Corrige el problema # 5623.
    - [bdubbs] - Actualización a shadow-4.17.2. Corrige el problema # 5625.
    - [bdubbs] - Actualización a setuptools-75.8.0. Corrige el problema #5 622.
    - [bdubbs] - Actualización a linux-6.12.9. Corrige el problema # 5620.
    - [bdubbs] - Actualización a gettext-0.23.1. Corrige el problema # 5619.
- 01/01/2025.
    - [renodr] - Actualización a libxcrypt-4.4.37. Corrige el problema # 5618.
    - [bdubbs] - Actualización a iana-etc-20241220. Corrige el problema # 5006.

    - [bdubbs] - Actualización a texinfo-7.2. Corrige el problema # 5616.
    - [bdubbs] - Actualización a sysvinit-3.12. Corrige el problema # 5615.
    - [bdubbs] - Actualización a shadow-4.17.1. Corrige el problema # 5617.
    - [bdubbs] - Actualización a procps-ng-4.0.5. Corrige el problema # 5611.
    - [bdubbs] - Actualización a meson-1.6.1. Corrige el problema # 5610.
    - [bdubbs] - Actualización a linux-6.12.7. Corrige el problema # 5613.
    - [bdubbs] - Actualización a kbd-2.7.1. Corrige el problema # 5608.
    - [bdubbs] - Actualización a jinja2-3.1.5 (Actualización de seguridad). Corrige el problema # 5614.
- 15/12/2024.
    - [bdubbs] - Actualización a vim-9.1.0927. Corrige el problema # 4500.
    - [bdubbs] - Actualización a iana-etc-20241206. Corrige el problema # 5006.
    - [bdubbs] - Actualización a systemd-257. Corrige el problema # 5559.
    - [bdubbs] - Actualización a Python-3.13.1 (Actualización de seguridad). Corrige el problema # 5605.
    - [bdubbs] - Actualización a libcap-2.73. Corrige el problema # 5604.
    - [bdubbs] - Actualización a linux-6.12.5. Corrige el problema # 5607.
    - [bdubbs] - Actualización a kbd-2.7. Corrige el problema # 5608.
    - [bdubbs] - Actualización a gettext-0.23. Corrige el problema # 5603.
- 01/12/2024.
    - [bdubbs] - Actualización a iana-etc-20241122. Corrige el problema # 5006.
    - [bdubbs] - Actualización a file-5.46. Corrige el problema # 5601.
    - [bdubbs] - Actualización a iproute2-6.12.0. Corrige el problema # 5597.
    - [bdubbs] - Actualización a libtool-2.5.4. Corrige el problema # 5598.
    - [bdubbs] - Actualización a linux-6.12.1. Corrige el problema # 5586.
    - [bdubbs] - Actualización a setuptools-75.6.0 (módulo de Python). Corrige el problema # 5599.
    - [bdubbs] - Actualización a wheel-0.45.1 (Módulo de Python). Corrige el problema # 5600.
- 15/11/2024.
    - [bdubbs] - Actualización a vim-9.1.0866. Corrige el problema # 4500.
    - [bdubbs] - Actualización a iana-etc-20241024. Corrige el problema # 5006.
    - [bdubbs] - Actualización a wheel-0.45.0 (Módulo de Python). Corrige el problema # 5593.
    - [bdubbs] - Actualización a setuptools-75.5.0 (Módulo de Python). Corrige el problema # 5595.
    - [bdubbs] - Actualización a linux-6.11.8. Corrige el problema # 5582.

    - [bdubbs] - Actualización a libcap-2.72. Corrige el problema # 5594.

- 08/11/2024.
    - [bdubbs] - Se añadió binutils-2.43.1-upstream_fix-1.patch. Corrige el problema # 5591.
    - [bdubbs] - Actualización a flit_core-3.10.1. Corrige el problema # 5589.
    - [bdubbs] - Actualización a expat-2.6.4. Corrige el problema # 5590.
- 25/10/2024.
    - [bdubbs] - Actualización a linux-6.11.6. Corrige el problema # 5588.
    - [bdubbs] - Actualización a libcap-2.71. Corrige el problema # 5584.
    - [bdubbs] - Actualización a setuptools-75.3.0. Corrige el problema # 5585.
    - [bdubbs] - Actualización a flit_core-3.10.0. Corrige el problema # 5587.
- 25/10/2024.
    - [bdubbs] - Actualización a iana-etc-20241015. Corrige el problema # 5006.
    - [bdubbs] - Actualización a vim-9.1.0813. Corrige el problema # 4500.
    - [bdubbs] - Actualización a xz-5.6.3. Corrige el problema # 5572.
    - [bdubbs] - Actualización a sysvinit-3.11. Corrige el problema # 5581.
    - [bdubbs] - Actualización a setuptools-75.2.0. Corrige el problema # 5577.
    - [bdubbs] - Actualización a Python3-3.13.0. Corrige el problema # 5575.
    - [bdubbs] - Actualización a openssl-3.4.0. Corrige el error # 5582.
    - [bdubbs] - Actualización a meson-1.6.0. Corrige el error # 5580.
    - [bdubbs] - Actualización a markupsafe-3.0.2. Corrige el error # 5576.
    - [bdubbs] - Actualización a linux-6.11.5. Corrige el error # 5574.
    - [bdubbs] - Actualización a less-668. Corrige el error # 5578.
    - [bdubbs] - Actualización a elfutils-0.192. Corrige el error # 5579.
- 03/10/2024.
    - [bdubbs] - Retorno a tcl8.6.15.
- 01/10/2024
    - [bdubbs] - Actualización a Python3-3.12.7. Corrige el problema # 5571.
    - [bdubbs] - Actualización a tcl9.0.0. Corrige el problema # 5570.
    - [bdubbs] - Actualización a linux-6.11.1. Corrige el problema # 5556.
    - [bdubbs] - Actualización a libtool-2.5.3. Corrige el problema # 5569.
    - [bdubbs] - Actualización a iproute2-6.11.0. Corrige el problema # 5561.

    - [bdubbs] - Actualización a bash-5.2.37. Corrige el problema # 5567.
    - [bdubbs] - Actualización a bc-7.0.3. Corrige el problema # 5568.
- 20/09/2024.
    - [bdubbs] - Actualización a vim-9.1.0738. Corrige el problema # 4500.
    - [bdubbs] - Actualización a texinfo-7.1.1. Corrige el problema # 5558.
    - [bdubbs] - Actualización a tcl8.6.15. Corrige el problema # 5562.
    - [bdubbs] - Actualización a sysklogd-2.6.2. Corrige el problema # 5557.
    - [bdubbs] - Actualización a setuptools-75.1.0. Corrige el problema # 5560.
    - [bdubbs] - Actualización a meson-1.5.2. Corrige el problema # 5566.
    - [bdubbs] - Actualización a iana-etc-20240912. Corrige el problema # 5006.
    - [bdubbs] - Actualización a gawk-5.3.1. Corrige el error # 5564.
    - [bdubbs] - Actualización a bc-7.0.2. Corrige el error # 5563.
- 07/09/2024.
    - [bdubbs] - Actualización a tzdata-2024b. Corrige el error # 5554.
    - [bdubbs] - Actualización a systemd-256.5. Corrige el error # 5551.
    - [bdubbs] - Actualización a setuptools-74.1.2. Corrige el error # 5546.
    - [bdubbs] - Actualización a python3-3.12.6. Corrige el error # 5555.
    - [bdubbs] - Actualización a openssl-3.3.2. Corrige el error # 5552.
    - [bdubbs] - Actualización a man-db-2.13.0. Corrige el error # 5550.
    - [bdubbs] - Actualización a linux-6.10.8. Corrige el error n.° 5545.
    - [bdubbs] - Actualización a libpipeline-1.5.8. Corrige el error # 5548.
    - [bdubbs] - Actualización a expat-2.6.3. Corrige el error # 5553.
    - [bdubbs] - Actualización a bc-7.0.1. Corrige el error # 5547.
- 01/09/2024.
    - [bdubbs] - Publicación de LFS-12.2.

1.4. Recursos

1.4.1. Preguntas frecuentes - FAQ

Si durante la compilación del sistema LFS encuentra algún error, tiene alguna pregunta o cree que hay una errata en el libro, consulte la lista de preguntas frecuentes en *https://www.linuxfromscratch.org/faq/*

1.4.2. Listas de correo

El servidor linuxfromscratch.org alberga varias listas de correo utilizadas para el desarrollo del proyecto LFS. Estas listas incluyen las principales de desarrollo y soporte, entre otras. Si no encuentra la respuesta a su problema en la página de preguntas frecuentes, el siguiente paso es buscar en las listas de correo en *https://www.linuxfromscratch.org/search.html*.

Para obtener información sobre las diferentes listas, cómo suscribirse, la ubicación de los archivos e información adicional, visite *https://www.linuxfromscratch.org/mail.html*.

1.4.3. IRC

Varios miembros de la comunidad LFS ofrecen asistencia a través de Internet Relay Chat (IRC). Antes de usar este soporte, asegúrese de que su pregunta no esté ya respondida en las preguntas frecuentes de LFS ni en los archivos de las listas de correo. Puede encontrar la red IRC en irc.libera.chat. El canal de soporte se llama #lfs-support.

1.4.4. Sitios espejo

El proyecto LFS cuenta con varios sitios espejo en todo el mundo para facilitar el acceso al sitio web y la descarga de los paquetes necesarios. Visite el sitio web de LFS para obtener una lista de los sitios espejo actuales.

https://www.linuxfromscratch.org/mirrors.html

1.4.5. Información de contacto

Dirija todas sus preguntas y comentarios a una de las listas de correo de LFS (ver arriba).

1.5. Ayuda

Nota

En caso de que haya encontrado un problema al compilar un paquete con la instrucción LFS, le recomendamos encarecidamente que no publique el problema directamente en el canal de soporte original antes de discutirlo a través de uno de los canales de soporte de LFS que se indican en la Sección 1.4, "Recursos". Hacerlo suele ser bastante ineficiente, ya que los desarrolladores originales rara vez están familiarizados con el procedimiento de compilación de LFS. Incluso si realmente ha encontrado un problema original, la comunidad LFS puede ayudarle a aislar la información solicitada por los desarrolladores originales y a elaborar un informe adecuado. Si necesita hacer una pregunta directamente a través de un canal de soporte, tenga en cuenta que muchos proyectos tienen canales de soporte separados del sistema de seguimiento de errores. Los informes de errores para hacer preguntas se consideran inválidos y pueden molestar a los desarrolladores de estos proyectos.

Si encuentra algún problema o pregunta mientras trabaja con este libro, consulte la página de preguntas frecuentes en – *FAQ* –. Las preguntas suelen estar ya respondidas allí. Si su pregunta no se responde en esa página, intente encontrar el origen del problema. La siguiente sugerencia le servirá de guía para la resolución de problemas: *https://www.linuxfromscratch.org/hints/downloads/files/errors.txt*.
Si no encuentra su problema en las preguntas frecuentes, busque en las listas de correo en *https://www.linuxfromscratch.org/search.html*. También contamos con una maravillosa comunidad de LFS dispuesta a ofrecer asistencia a través de las listas de correo e IRC (consulte la Sección 1.4, "*Recursos*" de este libro). Sin embargo, recibimos varias preguntas de soporte a diario, y muchas de ellas podrían haberse respondido fácilmente consultando las preguntas frecuentes o buscando primero en las listas de correo. Por lo tanto, para que podamos ofrecerle la mejor asistencia posible, primero debería investigar por su cuenta. Esto nos permitirá centrarnos en las necesidades de soporte más inusuales. Si sus búsquedas no dan una solución, incluya toda la información relevante (mencionada a continuación) en su solicitud de ayuda.

1.5.1. Aspectos a mencionar

Además de una breve explicación del problema, cualquier solicitud de ayuda debe incluir estos aspectos esenciales:

- La versión del libro que se está utilizando (en este caso, la 12.3)
- La distribución del host y la versión que se está utilizando para crear LFS
- El resultado del script de requisitos del sistema
- El paquete o la sección donde se encontró el problema
- El mensaje de error exacto o una descripción clara del problema
- Indique si se ha desviado del libro

Nota

Desviarse del libro no significa que no le ayudaremos. Al fin y al cabo, LFS se basa en preferencias personales.
Ser transparentes sobre cualquier cambio en el procedimiento establecido nos ayuda a evaluar y determinar las posibles causas de su problema.

1.5.2. Problemas con el script de configuración

Si algo falla al ejecutar el script de **configuración**, revise el archivo config.log. Este archivo puede contener errores encontrados durante la **configuración** que no se mostraron en pantalla. Incluya las líneas relevantes si necesita solicitar ayuda.

1.5.3. Problemas de compilación

Tanto la salida en pantalla como el contenido de varios archivos son útiles para determinar la causa de los problemas de compilación.

La salida en pantalla del script de **configuración** y la ejecución de **make** pueden ser útiles. No es necesario incluir la salida completa, pero sí toda la información relevante. A continuación, se muestra un ejemplo del tipo de información que se debe incluir en la salida en pantalla de **make**.

```
gcc -D ALIASPATH=\"/mnt/lfs/usr/share/locale:.\"
-D LOCALEDIR=\"/mnt/lfs/usr/share/locale\"
-D LIBDIR=\"/mnt/lfs/usr/lib\"
-D INCLUDEDIR=\"/mnt/lfs/usr/include\" -D HAVE_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: En la función `load_too_high':
/lfs/tmp/make-3.79.1/job.c:1565: referencia indefinida
a `getloadavg'
collect2: ld devolvió 1 estado de salida
make[2]: *** [make] Error 1
make[2]: Saliendo del directorio `/lfs/tmp/make-3.79.1'
make[1]: *** [all-recursive] Error 1
make[1]: Saliendo del directorio `/lfs/tmp/make-3.79.1'
make: *** [all-recursive-am] Error 2
```

En este caso, muchos usuarios solo incluirían la sección inferior:

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

Esta información no es suficiente para diagnosticar el problema, ya que solo indica que algo salió mal, no qué salió mal. La sección completa, como en el ejemplo anterior, es lo que se debe guardar, ya que incluye el comando ejecutado y todos los mensajes de error asociados.

Hay un excelente artículo sobre cómo pedir ayuda en internet disponible en:

http://catb.org/~esr/faqs/smart-questions.html

Lea este documento y siga las sugerencias. Esto aumentará las probabilidades de obtener la ayuda que necesita.

Parte II. Preparación para la construcción

Capítulo 2. Preparación del Sistema Host

2.1. Introducción

En este capítulo, se comprueban las herramientas del host necesarias para compilar LFS y, si es necesario, se instalan. A continuación, se prepara una partición que alojará el sistema LFS. Crearemos la partición, crearemos un sistema de archivos en ella y la montaremos.

2.2. Requisitos del Sistema Host

2.2.1. Hardware

Los editores de LFS recomiendan que la CPU del sistema tenga al menos cuatro núcleos y al menos 8 GB de memoria. Los sistemas antiguos que no cumplan estos requisitos seguirán funcionando, pero el tiempo de compilación de los paquetes será significativamente mayor que el documentado.

2.2.2. Software

Su Sistema Host debe tener el siguiente software con las versiones mínimas indicadas. Esto no debería ser un problema para la mayoría de las distribuciones modernas de Linux. Tenga en cuenta también que muchas distribuciones colocan las cabeceras del software en paquetes separados, a menudo con la forma <package-name>-devel o <package-name>-dev. Asegúrese de instalarlos si su distribución los proporciona.

Versiones anteriores de los paquetes de software mencionados podrían funcionar, pero no se han probado.

- **Bash-3.2** (/bin/sh debe ser un enlace simbólico o físico a bash).
- **Binutils-2.13.1** (No se recomiendan versiones posteriores a la 2.44, ya que no se han probado).
- **Bison-2.7.** (/usr/bin/yacc debe ser un enlace a bison o un pequeño script que l o ejecute).
- **Coreutils-8.1**.
- **Diffutils-2.8.1**.
- **Findutils-4.2.3**1.
- **Gawk-4.0.1** (/usr/bin/awk debe ser un enlace a gawk).
- **GCC-5.2.** Incluido el compilador de C++, g++ (No se recomiendan versiones posteriores a la 14.2.0, ya que no se han probado). Las bibliotecas estándar de C y C++ (con sus cabeceras) también deben estar presentes para que el compilador de C++ pueda compilar programas alojados.
- **Grep-2.5.1a**.
- **Gzip-1.3.12**.
- **Kernel Linux-5.4**.

El requisito de la versión del kernel se debe a que la especificamos al compilar glibc en los capítulos 5 y 8, por lo que las soluciones alternativas para kernels antiguos no están habilitadas y el glibc compilado es ligeramente más rápido y más pequeño. A diciembre de 2024, la versión 5.4 era la versión más antigua del kernel que aún recibía soporte de los desarrolladores. Algunas versiones del kernel anteriores a la 5.4 podrían seguir recibiendo soporte de equipos externos, pero no se consideran versiones oficiales del kernel original; consulte https://kernel.org/category/releases.html para obtener más información.

Si el kernel del **Host** es anterior a la 5.4, deberá reemplazarlo por una versión más actualizada. Hay dos maneras de hacerlo. Primero, verifique si su proveedor de Linux ofrece un paquete de kernel 5.4 o posterior. De ser así, puede instalarlo. Si su proveedor no ofrece un paquete de kernel aceptable o prefiere no instalarlo, puede compilar un kernel usted mismo. Las instrucciones para compilar el kernel y configurar el gestor de arranque (suponiendo que el host usa GRUB) se encuentran en el Capítulo 10.

Requerimos que el kernel del host sea compatible con la pseudoterminal (PTY) de UNIX 98. Debe estar habilitada en todas las distribuciones de escritorio o servidor que incluyan Linux 5.4 o un kernel posterior. Si está compilando un kernel personalizado para el host, asegúrese de que CONFIG_UNIX98_PTYS esté establecida en ¨y¨ (de **YES**) en la configuración del kernel.

- **M4-1.4.10**
- **Make-4.0**
- **Patch-2.5.4**
- **Perl-5.8.8**
- **Python-3.4**
- **Sed-4.1.5**
- **Tar-1.22**
- **Texinfo-5.0**
- **Xz-5.0.0**

Importante

Tenga en cuenta que los enlaces simbólicos mencionados anteriormente son necesarios para construir un sistema LFS siguiendo las instrucciones de este libro. Los enlaces simbólicos que apuntan a otro software (como dash, mawk, etc.) pueden funcionar, pero el equipo de desarrollo de LFS no los ha probado ni soportado, y podrían requerir modificaciones de las instrucciones o parches adicionales para algunos paquetes.

Para comprobar si su Sistema Host tiene todas las versiones adecuadas y la capacidad de compilar programas, cree el script version-check.sh, el cual ejecutará una serie de comandos de verificación:

Script version-check.sh

```
cat > version-check.sh << "EOF"
#!/bin/bash
# Un script para listar los números de versión de herramientas de desarrollo
críticas.

# Si tiene herramientas instaladas en otros directorios, ajuste PATH aquí
# Y también en ~lfs/.bashrc (sección 4.4).

LC_ALL=C
PATH=/usr/bin:/bin

bail() { echo "FATAL: $1"; exit 1; }
grep --version > /dev/null 2> /dev/null || bail "grep no funciona"
sed '' /dev/null || bail "sed no funciona"
sort   /dev/null || bail "sort no funciona"

ver_check()
{
   if ! type -p $2 &>/dev/null
   then
     echo "ERROR: No se puede encontrar $2 ($1)"; return 1;
fi
v=$($2 --version 2>&1 | grep -E -o '[0-9]+\.[0-9\.]+[a-z]*' | head -n1)
if printf '%s\n' $3 $v | sort --version-sort --check &>/dev/null
then
  printf "OK: %-9s %-6s >= $3\n" "$1" "$v"; return 0;
else
```

```
      printf "ERROR: %-9s es DEMASIADO ANTIGUO ($3 o posterior es requerido)\n"
"$1";
     return 1;
   fi
}

ver_kernel()
{
   kver=$(uname -r | grep -E -o '^[0-9\.]+')
   if printf '%s\n' $1 $kver | sort --version-sort --check &>/dev/null
   then
     printf "OK: El Kerne Linuxl $kver >= $1\n"; return 0;
   else
      printf "ERROR: El Kernel Linux ($kver) es DEMASIADO ANTIGUO ($1 o posterior
es requerido)\n" "$kver";
    return 1;
   fi
}

# Coreutils va primero porque --version-sort necesita Coreutils >= 7.0
ver_check Coreutils    sort 8.1 || bail "Coreutils demasiado antiguo, detente"
ver_check Bash         bash 3.2
ver_check Binutils     ld 2.13.1
ver_check Bison        bison 2.7
ver_check Diffutils    diff 2.8.1
ver_check Findutils    find 4.2.31
ver_check Gawk         gawk 4.0.1
ver_check GCC          gcc 5.2
ver_check "GCC (C++)"  g++ 5.2
ver_check Grep         grep 2.5.1a
ver_check Gzip         gzip 1.3.12
ver_check M4           m4 1.4.10
ver_check Make         make 4.0
ver_check Patch        patch 2.5.4
ver_check Perl         perl 5.8.8
ver_check Python       python3 3.4
ver_check Sed          sed 4.1.5
ver_check Tar          tar 1.22
ver_check Texinfo      texi2any 5.0
ver_check Xz           xz 5.0.0
ver_kernel 5.4

if mount | grep -q 'devpts on /dev/pts' && [ -e /dev/ptmx ]
then echo "OK:  El kernel Linux es compatible con UNIX 98 PTY";
else echo "ERROR:  El kernel Linux NO es compatible con UNIX 98 PTY"; fi

alias_check()
{
   if $1 --version 2>&1 | grep -qi $2
   then printf "OK: %-4s es $2\n" "$1";
   else printf "ERROR: %-4s NO es $2\n" "$1"; fi
}
echo "Alias:"
alias_check awk GNU
alias_check yacc Bison
alias_check sh Bash

echo "Comprobación del compilador:"
if printf "int main(){}" | g++ -x c++ -
```

```
then echo "OK: g++ Funciona";
else echo "ERROR: g++ NO Funciona"; fi
rm -f a.out

if [ "$(nproc)" = "" ]; then
   echo "ERROR: nproc no está disponible o produce una salida vacía"
else
   echo "OK: nproc informa que $(nproc) núcleos lógicos están disponibles"
fi
EOF

bash version-check.sh
```

2.3. Compilación de LFS por etapas

LFS está diseñado para compilarse en una sola sesión. Es decir, las instrucciones asumen que el sistema no se apagará durante el proceso. Esto no significa que el sistema deba compilarse de una sola vez. El problema radica en que ciertos procedimientos deben repetirse después de reiniciar al reanudar LFS en diferentes puntos.

2.3.1. Capítulos 1-4

Estos capítulos ejecutan comandos en el sistema host. Al reiniciar, asegúrese de una cosa:

• Los procedimientos realizados como usuario root después de la Sección 2.4 deben tener la variable de entorno LFS configurada *PARA EL USUARIO ROOT*.

2.3.2. Capítulos 5-6

• La partición /mnt/lfs debe estar montada.

• Estos dos capítulos deben ejecutarse como usuario lfs. Debe ejecutarse el comando **su - lfs** antes de realizar cualquier tarea en estos capítulos. De lo contrario, corre el riesgo de instalar paquetes en el host y potencialmente dejarlo inutilizable.

• Los procedimientos de las Instrucciones Generales de Compilación son fundamentales. Si tiene alguna duda sobre la instalación correcta de un paquete, asegúrese de que se haya eliminado el archivo tar previamente expandido, vuelva a extraer el paquete y complete todas las instrucciones de esa sección.

2.3.3. Capítulos 7–10

• La partición /mnt/lfs debe estar montada.

• Algunas operaciones, desde “Preparación de los sistemas de archivos virtuales del kernel” hasta “Ingreso al entorno chroot”, deben realizarse como usuario root, con la variable de entorno LFS configurada para el usuario root.

• Al ingresar al entorno chroot, la variable de entorno LFS debe estar configurada para el usuario root. La variable LFS no se utiliza después de ingresar al entorno chroot.

• Los sistemas de archivos virtuales deben estar montados. Esto puede hacerse antes o después de ingresar al entorno chroot, cambiando a una terminal virtual del host y, como usuario root, ejecutando los comandos de la Sección 7.3.1, “Montaje y preparación de /dev” y la Sección 7.3.2, “Montaje de sistemas de archivos virtuales del kernel”.

2.4. Creación de una nueva partición

Como la mayoría de los sistemas operativos, LFS suele instalarse en una partición dedicada. El enfoque recomendado para construir un sistema LFS es usar una partición vacía disponible o, si dispone de suficiente espacio sin particionar, crear una.

Un sistema mínimo requiere una partición de unos 10 gigabytes (GB). Esto es suficiente para almacenar todos los archivos tar de origen y compilar los paquetes. Sin embargo, si el sistema LFS se pretende que sea el sistema Linux principal, probablemente se instalará software adicional, lo que requerirá espacio adicional. Una partición de 30 GB es un tamaño razonable para permitir el crecimiento.

El sistema LFS en sí no ocupará tanto espacio. Gran parte de este requisito consiste en proporcionar suficiente almacenamiento temporal libre, así como en añadir capacidades adicionales una vez finalizado el LFS. Además, la compilación de paquetes puede requerir mucho espacio en disco, que se recuperará tras la instalación del paquete.

Dado que no siempre hay suficiente memoria de acceso aleatorio (RAM) disponible para los procesos de compilación, es recomendable usar una pequeña partición de disco como espacio de intercambio. El núcleo la utiliza para almacenar datos que rara vez se utilizan y dejar más memoria disponible para los procesos activos. La partición de intercambio de un sistema LFS puede ser la misma que la utilizada por el sistema host, en cuyo caso no es necesario crear otra.

Inicie un programa de particionado de discos, como **cfdisk** o **fdisk**, con una opción de línea de comandos que indique el nombre del disco duro donde se creará la nueva partición; por ejemplo, /dev/sda para la unidad de disco principal. Cree una partición nativa de Linux y una partición de intercambio (swap), si es necesario. Consulte *cfdisk(8)* o *fdisk(8)* si aún no sabe cómo usar estos programas.

Nota

Para usuarios experimentados, existen otros esquemas de particionado. El nuevo sistema LFS puede estar en una matriz RAID por software o en un volumen lógico LVM. Sin embargo, algunas de estas opciones requieren un archivo *initramfs*, que es un tema avanzado. Estas metodologías de particionado no se recomiendan para usuarios nuevos de LFS.

Recuerde la designación de la nueva partición (p. ej., sda5). En este libro, la llamaremos partición LFS. Recuerde también la designación de la partición de intercambio. Estos nombres serán necesarios más adelante para el archivo /etc/fstab.

2.4.1. Otros problemas de particiones

Las solicitudes de asesoramiento sobre particionamiento del sistema suelen publicarse en las listas de correo de LFS. Este es un tema muy subjetivo.

La configuración predeterminada para la mayoría de las distribuciones es usar todo el disco duro, con la excepción de una pequeña partición de intercambio (swap). Esto no es óptimo para LFS por varias razones. Reduce la flexibilidad, dificulta el intercambio de datos entre múltiples distribuciones o

compilaciones de LFS, hace que las copias de seguridad consuman más tiempo y puede desperdiciar espacio en disco debido a una asignación ineficiente de las estructuras del sistema de archivos.

2.4.1.1. La partición raíz

Una partición raíz de LFS (no confundir con el directorio /root) de veinte gigabytes es una buena opción para la mayoría de los sistemas. Proporciona suficiente espacio para compilar LFS y la mayor parte de BLFS, pero es lo suficientemente pequeña como para permitir la creación fácil de múltiples particiones para experimentación.

2.4.1.2. La partición de intercambio

La mayoría de las distribuciones crean automáticamente una partición de intercambio. Generalmente, el tamaño recomendado de la partición de intercambio es aproximadamente el doble de la RAM física; sin embargo, esto rara vez se necesita. Si el espacio en disco es limitado, mantenga la partición de intercambio a dos gigabytes y supervise la cantidad de intercambio de disco.

Si desea utilizar la función de hibernación (suspensión a disco) de Linux, esta escribe el contenido de la RAM en la partición de intercambio antes de apagar el equipo. En este caso, el tamaño de la partición de intercambio debe ser al menos tan grande como la RAM instalada en el sistema.

El intercambio nunca es recomendable. En el caso de los discos duros mecánicos, generalmente se puede saber si un sistema está intercambiando simplemente escuchando la actividad del disco y observando cómo reacciona el sistema a los comandos. Con un SSD no se podrá oír el intercambio, pero se puede saber cuánto espacio de intercambio se está utilizando ejecutando los programas principales o gratuitos. Se debe evitar el uso de un SSD como partición de intercambio siempre que sea posible. La primera reacción al intercambio debe ser comprobar si hay un comando irrazonable, como intentar editar un archivo de cinco gigabytes. Si el intercambio se convierte en algo habitual, la mejor solución es adquirir más RAM para el sistema.

2.4.1.3. La partición de la BIOS de Grub

Si el *disco de arranque* se ha particionado con una tabla de particiones GUID (GPT), se debe crear una partición pequeña, normalmente de 1 MB, si no existe. Esta partición no está formateada, pero debe estar disponible para que GRUB la utilice durante la instalación del gestor de arranque. Esta partición normalmente se etiquetará como 'BIOS Boot' si se usa **fdisk** o tendrá el código *EF02* si se usa el comando **gdisk**.

Nota

La partición Grub Bios debe estar en la unidad que la BIOS utiliza para arrancar el sistema. Esta no es necesariamente la unidad que contiene la partición raíz LFS. Los discos de un sistema pueden usar diferentes tipos de tablas de particiones. La necesidad de la partición Grub Bios depende únicamente del tipo de tabla de particiones del disco de arranque.

2.4.1.4. Particiones de conveniencia

Existen otras particiones que no son necesarias, pero que deben tenerse en cuenta al diseñar la distribución del disco. La siguiente lista no es exhaustiva, sino una guía.

• /boot - Muy recomendable. Utilice esta partición para almacenar kernels y otra información de arranque. Para minimizar posibles problemas de arranque con discos más grandes, crea esta partición como la primera partición física en tu primer disco. Un tamaño de partición de 200 megabytes es suficiente.

• /boot/efi - La partición del sistema EFI, necesaria para arrancar el sistema con UEFI. Consulte la página de BLFS para obtener más información.

• /home - Muy recomendable. Comparte tu directorio personal y la personalización de usuario entre varias distribuciones o compilaciones LFS. El tamaño suele ser bastante grande y depende del espacio disponible en disco.

• /usr - En LFS, /bin, /lib y /sbin son enlaces simbólicos a sus equivalentes en /usr. Por lo tanto, /usr contiene todos los binarios necesarios para el funcionamiento del sistema. Para LFS, normalmente no se necesita una partición independiente para /usr. Si la creas de todos modos, deberías crear una partición lo suficientemente grande como para que quepan todos los programas y bibliotecas del sistema. La partición raíz puede ser muy pequeña (quizás de solo un gigabyte) en esta configuración, por lo que es adecuada para un cliente ligero o una estación de trabajo sin disco (donde /usr se monta desde un servidor remoto). Sin embargo, debes tener en cuenta que se necesitará un initramfs (no incluido en LFS) para arrancar un sistema con una partición /usr independiente.

• /opt - Este directorio es especialmente útil para BLFS, donde se pueden instalar varios paquetes grandes como KDE o Texlive sin incrustar los archivos en la jerarquía /usr. Si se utiliza, de 5 a 10 gigabytes suele ser suficiente.

• /tmp - Una partición /tmp independiente es poco común, pero útil si se configura un cliente ligero. Esta partición, si se utiliza, no suele necesitar superar un par de gigabytes. Si tiene suficiente RAM, puede montar un *tmpfs* en /tmp para agilizar el acceso a los archivos temporales.

• /usr/src - Esta partición es muy útil para proporcionar una ubicación donde almacenar archivos fuente de BLFS y compartirlos entre compilaciones de LFS. También se puede utilizar como ubicación para compilar paquetes de BLFS. Una partición razonablemente grande de 30 a 50 gigabytes proporciona suficiente espacio.

Cualquier partición independiente que desee que se monte automáticamente al iniciar el sistema debe especificarse en el archivo /etc/fstab. Los detalles sobre cómo especificar particiones se tratarán en la Sección 10.2, “Creación del archivo /etc/fstab”.

2.5. Creación de un sistema de archivos en la partición

Una partición es simplemente un conjunto de sectores en una unidad de disco, delimitados por los límites establecidos en una tabla de particiones. Antes de que el sistema operativo pueda usar una partición para almacenar archivos, esta debe formatearse para contener un sistema de archivos, que generalmente consta de una etiqueta, bloques de directorio, bloques de datos y un esquema de indexación para localizar un archivo específico cuando se necesite. El sistema de archivos también ayuda al sistema operativo a controlar el espacio libre en la partición, a reservar los sectores necesarios cuando se crea un nuevo archivo o se extiende uno existente, y a reciclar los segmentos de datos libres creados al eliminar archivos. También puede proporcionar soporte para redundancia de datos y recuperación de errores.

LFS puede usar cualquier sistema de archivos reconocido por el kernel de Linux, pero los tipos más comunes son ext3 y ext4. La elección del sistema de archivos adecuado puede ser compleja; Depende de las características de los archivos y del tamaño de la partición. Por ejemplo:

ext2
: es adecuado para particiones pequeñas que se actualizan con poca frecuencia, como /boot.

ext3
: es una actualización de ext2 que incluye un diario para ayudar a recuperar el estado de la partición en caso de un apagado incorrecto.

  Se utiliza comúnmente como sistema de archivos de propósito general.

ext4
: es la última versión de la familia de sistemas de archivos ext. Ofrece varias funciones nuevas, como marcas de tiempo de nanosegundos, creación y uso de archivos muy grandes (hasta 16 TB) y mejoras de velocidad.

Otros sistemas de archivos, como FAT32, NTFS, JFS y XFS, son útiles para fines especializados. Puede encontrar más información sobre estos sistemas de archivos y muchos otros en:

https://en.wikipedia.org/wiki/Comparison_of_file_systems.

LFS asume que el sistema de archivos raíz (/) es de tipo ext4. Para crear un sistema de archivos ext4 en la partición LFS, ejecute el siguiente comando:

```
mkfs -v -t ext4 /dev/<xxx>
```

Reemplace <*xxx*> con el nombre de la partición LFS.

Si utiliza una partición de intercambio existente, no es necesario formatearla. Si se creó una nueva partición de intercambio, deberá inicializarse con este comando:

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

Reemplace <*yyy*> con el nombre de la partición de *intercambio*.

2.6. Configuración de la variable $LFS y Umask

A lo largo de este libro, la variable de entorno LFS se utilizará varias veces. Debe asegurarse de que esta variable esté siempre definida durante el proceso de compilación de LFS. Debe configurarse con el nombre del directorio donde compilará su sistema LFS; usaremos */mnt/lfs* como ejemplo, pero puede elegir el nombre de directorio que desee. Si compila LFS en una partición separada, este directorio será el punto de montaje de la partición. Seleccione una ubicación de directorio y configure la variable con el siguiente comando:

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

Tener esta variable configurada es beneficioso, ya que comandos como **mkdir -v $LFS/tools** se pueden escribir literalmente. El shell reemplazará automáticamente "$LFS" por "/mnt/lfs" (o el valor que se haya configurado para la variable) al procesar la línea de comandos.

Ahora configure la máscara de creación de modo de archivo (umask) en *022* en caso de que la distribución del host use un valor predeterminado diferente:

```
umask 022
```

Configurar la umask en 022 garantiza que los archivos y directorios recién creados solo puedan ser escritos por su propietario, pero que cualquier persona pueda leer y buscar (solo directorios) (suponiendo que la llamada al sistema *open(2)* utilice los modos predeterminados, los nuevos archivos tendrán el modo de permiso 644 y los directorios el modo 755). Una configuración predeterminada demasiado permisiva puede dejar vulnerabilidades de seguridad en el sistema LFS, y una configuración predeterminada demasiado restrictiva puede causar problemas extraños al crear o usar el sistema LFS.

Precaución

No olvide comprobar que *LFS* esté configurado y que la máscara de usuario (umask) esté configurada en *022* al salir y volver a entrar en el entorno de trabajo actual (por ejemplo, al ejecutar un comando su como *root* u otro usuario). Compruebe que la variable LFS esté configurada correctamente con:

```
    echo $LFS
```

Asegúrese de que la salida muestre la ruta a la ubicación de compilación de su sistema LFS, que es */mnt/lfs* si se siguió el ejemplo proporcionado.
Compruebe que la máscara de usuario (umask) esté configurada correctamente con:

```
    umask
```

La salida puede ser *0022* o *022* (el número de ceros a la izquierda depende de la distribución del host).
Si alguno de los resultados de estos dos comandos es incorrecto, utilice el comando indicado anteriormente en esta página para configurar $LFS con el nombre de directorio correcto y establecer umask en *022*.

Nota

Una forma de garantizar que la variable LFS y umask siempre estén configuradas correctamente es editar el archivo *.bash_profile*, tanto en su directorio personal como en */root/.bash_profile*, e introducir los comandos **export** y **umask** mencionados anteriormente. Además, el shell especificado en el archivo /etc/passwd para todos los usuarios que necesitan la variable *LFS* debe ser bash para garantizar que el archivo *.bash_profile* se incorpore al proceso de inicio de sesión.

Otra consideración es el método utilizado para iniciar sesión en el sistema host. Si se inicia sesión a través de un gestor de pantalla gráfica, el archivo *.bash_profile* del usuario no suele utilizarse al iniciar una terminal virtual. En este caso, añada los comandos al archivo *.bashrc* para el usuario y el usuario *root*. Además, algunas distribuciones utilizan una prueba "if" y no ejecutan las instrucciones *.bashrc* restantes para una invocación bash no interactiva. Asegúrese de colocar los comandos antes de la prueba para uso no interactivo.

2.7. Montaje de la nueva partición

Una vez creado el sistema de archivos, la partición debe montarse para que el sistema host pueda acceder a ella. Este libro asume que el sistema de archivos está montado en el directorio especificado por la variable de entorno LFS descrita en la sección anterior.
En sentido estricto, no se puede "montar una partición". Se monta el *sistema de archivos* integrado en esa partición. Sin embargo, como una sola partición no puede contener más de un sistema de archivos, a menudo se habla de la partición y el sistema de archivos asociado como si fueran uno solo. Cree el punto de montaje y monte el sistema de archivos LFS con estos comandos:

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
```

Reemplace <*xxx*> con el nombre de la partición LFS.

Si utiliza varias particiones para LFS (por ejemplo, una para / y otra para /home), móntelas así:

```
mkdir -pv $LFS
mount -v -t ext4 /dev/<xxx> $LFS
mkdir -v $LFS/home
mount -v -t ext4 /dev/<yyy> $LFS/home
```

Reemplace <*xxx*> y <*yyy*> con los nombres de partición correspondientes.

Configure el propietario y el modo de permisos del directorio `$LFS` (es decir, el directorio raíz del sistema de archivos recién creado para el sistema LFS) como *root* y *755* en caso de que la distribución del host se haya configurado para usar una configuración predeterminada diferente para **mkfs**:

```
chown root:root $LFS
chmod 755 $LFS
```

Asegúrese de que esta nueva partición no se monte con permisos demasiado restrictivos (como las opciones *nosuid* o *nodev*). Ejecute el comando **mount** sin parámetros para ver qué opciones están configuradas para la partición LFS montada.

Si *nosuid* o *nodev* están configurados, la partición debe volver a montarse.

Advertencia

Las instrucciones anteriores presuponen que no reiniciará el equipo durante el proceso LFS. Si apaga el sistema, deberá volver a montar la partición LFS cada vez que reinicie el proceso de compilación o modificar el archivo */etc/fstab* del sistema host para que se vuelva a montar automáticamente al reiniciar. Por ejemplo, puede agregar esta línea a su archivo */etc/fstab*:

```
/dev/<xxx>  /mnt/lfs ext4    defaults     1    1
```

Si utiliza particiones opcionales adicionales, asegúrese de agregarlas también.

Si utiliza una partición de *intercambio* (*swap*), asegúrese de que esté habilitada con el comando **swapon**:

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

Reemplace <zzz> con el nombre de la partición de intercambio.

Ahora que la nueva partición LFS está lista para su uso, es hora de descargar los paquetes.

Capítulo 3. Paquetes y parches

3.1. Introducción

Este capítulo incluye una lista de paquetes que deben descargarse para compilar un sistema Linux básico. Los números de versión indicados corresponden a versiones del software que se sabe que funcionan, y este libro se basa en su uso. Recomendamos encarecidamente no utilizar versiones diferentes, ya que los comandos de compilación de una versión podrían no funcionar con otra, a menos que la versión diferente se especifique en una errata de LFS o un aviso de seguridad. Las versiones más recientes del paquete también pueden presentar problemas que requieran soluciones alternativas. Estas soluciones se desarrollarán y estabilizarán en la versión de desarrollo del libro.

Para algunos paquetes, el archivo comprimido de la versión y el archivo comprimido de la instantánea del repositorio (Git o SVN) para esa versión pueden publicarse con nombres de archivo similares o incluso idénticos. Sin embargo, el archivo comprimido de la versión puede contener algunos archivos esenciales a pesar de no estar almacenados en el repositorio (por ejemplo, un script de **configuración** generado por **autoconf**), además del contenido de la instantánea del repositorio correspondiente. El libro utiliza archivos comprimidos de la versión siempre que sea posible. Usar una instantánea del repositorio en lugar del archivo comprimido de la versión especificado por el libro causará problemas.

Es posible que las ubicaciones de descarga no siempre sean accesibles. Si la ubicación de descarga ha cambiado desde la publicación de este libro, Google (*https://www.google.com/*) ofrece un buscador útil para la mayoría de los paquetes. Si la búsqueda no funciona, pruebe con otro método de descarga:

https://www.linuxfromscratch.org/lfs/mirrors.html#files.

Los paquetes y parches descargados deberán almacenarse en un lugar accesible durante toda la compilación. También se requiere un directorio de trabajo para descomprimir el código fuente y compilarlo. *$LFS/sources* puede usarse tanto para almacenar los archivos tar y los parches como para el directorio de trabajo. Al usar este directorio, los elementos necesarios se ubicarán en la partición LFS y estarán disponibles durante todas las etapas del proceso de compilación.

Para crear este directorio, ejecute el siguiente comando como usuario *root* antes de iniciar la descarga:

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

Configure este directorio como permanente y con permisos de escritura. "Permanente" significa que, incluso si varios usuarios tienen permiso de escritura en un directorio, solo el propietario de un archivo puede eliminarlo dentro de un directorio permanente. El siguiente comando habilitará los modos de escritura y permanente:

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

Hay varias maneras de obtener todos los paquetes y parches necesarios para compilar LFS:

• Los archivos se pueden descargar individualmente como se describe en las dos secciones siguientes.

• Para las versiones estables del libro, se puede descargar un archivo comprimido con todos los archivos necesarios desde uno de los sitios espejo listados en:

https://www.linuxfromscratch.org/mirrors.html#files.

• Los archivos se pueden descargar usando wget y una lista wget como se describe a continuación. Para descargar todos los paquetes y parches usando *wget-list* como entrada del comando **wget**, utilice:

```
wget --input-file=wget-list --continue --directory-prefix=$LFS/sources
```

Además, a partir de LFS-7.0, existe un archivo independiente, *md5sums*, que permite verificar la disponibilidad de todos los paquetes correctos antes de continuar. Coloque ese archivo en *$LFS/sources* y ejecute:

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

Esta comprobación puede realizarse después de recuperar los archivos necesarios con cualquiera de los métodos mencionados anteriormente.

Si los paquetes y parches se descargan como un usuario no-root, estos archivos serán propiedad del usuario. El sistema de archivos registra al propietario por su UID, y el UID de un usuario normal en la distribución host no se asigna en LFS. Por lo tanto, los archivos permanecerán bajo un UID sin nombre en el sistema LFS final. Si no asigna el mismo UID a su usuario en el sistema LFS, cambie la propiedad de estos archivos a root para evitar este problema:

```
chown root:root $LFS/sources/*
```

3.2. Todos los paquetes

Nota

Lea los avisos de seguridad antes de descargar paquetes para determinar si debe usar una versión más reciente de algún paquete y así evitar vulnerabilidades de seguridad.

Las fuentes originales pueden eliminar versiones anteriores, especialmente si estas contienen una vulnerabilidad de seguridad. Si no puede acceder a una de las URL a continuación, lea primero los avisos de seguridad para determinar si debe usar una versión más reciente (con la vulnerabilidad corregida). Si no es así, intenta descargar el paquete eliminado desde un servidor espejo. Aunque es posible descargar una versión anterior desde un servidor espejo incluso si esta se ha eliminado debido a una vulnerabilidad, no es recomendable usar una versión vulnerable al compilar el sistema.

Descargue u obtenga los siguientes paquetes:

• Acl (2.3.2) - 363 KB:
Página principal: https://savannah.nongnu.org/projects/acl
Descarga: https://download.savannah.gnu.org/releases/acl/acl-2.3.2.tar.xz
Suma MD5: 590765dee95907dbc3c856f7255bd669
• Attr (2.5.2) - 484 KB:
Página principal: https://savannah.nongnu.org/projects/attr
Descarga: https://download.savannah.gnu.org/releases/attr/attr-2.5.2.tar.gz
Suma MD5: 227043ec2f6ca03c0948df5517f9c927

• Autoconf (2.72) - 1360 KB:
Página principal: https://www.gnu.org/software/autoconf/
Descarga: https://ftp.gnu.org/gnu/autoconf/autoconf-2.72.tar.xz
Suma MD5: 1be79f7106ab6767f18391c5e22be701

• Automake (1.17) - 1614 KB:
Página principal: https://www.gnu.org/software/automake/
Descarga: https://ftp.gnu.org/gnu/automake/automake-1.17.tar.xz
Suma MD5: 7ab3a02318fee6f5bd42adfc369abf10

• Bash (5.2.37) – 10.868 KB:
Página principal: https://www.gnu.org/software/bash/
Descarga: https://ftp.gnu.org/gnu/bash/bash-5.2.37.tar.gz
Suma MD5: 9c28f21ff65de72ca329c1779684a972

• Bc (7.0.3) - 464 KB:
Página principal: https://git.gavinhoward.com/gavin/bc
Descarga: https://github.com/gavinhoward/bc/releases/download/7.0.3/bc-7.0.3.tar.xz
Suma MD5: ad4db5a0eb4fdbb3f6813be4b6b3da74

• Binutils (2.44) - 26.647 KB:
Página principal: https://www.gnu.org/software/binutils/
Descarga: https://sourceware.org/pub/binutils/releases/binutils-2.44.tar.xz
Suma MD5: 49912ce774666a30806141f106124294

• Bison (3.8.2) - 2.752 KB:
Página principal: https://www.gnu.org/software/bison/
Descarga: https://ftp.gnu.org/gnu/bison/bison-3.8.2.tar.xz
Suma MD5: c28f119f405a2304ff0a7ccdcc629713

• **Bzip2 (1.0.8) - 792 KB:**
Descarga: https://www.sourceware.org/pub/bzip2/bzip2-1.0.8.tar.gz
Suma MD5: 67e051268d0c475ea773822f7500d0e5

• **Check (0.15.2) - 760 KB:**
Página principal: https://libcheck.github.io/check
Descarga: https://github.com/libcheck/check/releases/download/0.15.2/check-0.15.2.tar.gz
Suma MD5: 50fcafcecde5a380415b12e9c574e0b2

• **Coreutils (9.6) – 5.991 KB:**
Página principal: https://www.gnu.org/software/coreutils/
Descarga: https://ftp.gnu.org/gnu/coreutils/coreutils-9.6.tar.xz
Suma MD5: 0ed6cc983fe02973bc98803155cc1733

• **DejaGNU (1.6.3) - 608 KB:**
Página principal: https://www.gnu.org/software/dejagnu/
Descarga: https://ftp.gnu.org/gnu/dejagnu/dejagnu-1.6.3.tar.gz
Suma MD5: 68c5208c58236eba447d7d6d1326b821

• **Diffutils (3.11) – 1.881 KB:**
Página principal: https://www.gnu.org/software/diffutils/
Descarga: https://ftp.gnu.org/gnu/diffutils/diffutils-3.11.tar.xz
Suma MD5: 75ab2bb7b5ac0e3e10cece85bd1780c2

• **E2fsprogs (1.47.2) – 9.763 KB:**
Página principal: https://e2fsprogs.sourceforge.net/
Descarga: https://downloads.sourceforge.net/project/e2fsprogs/e2fsprogs/v1.47.2/e2fsprogs-1.47.2.tar.gz
Suma MD5: 752e5a3ce19aea060d8a203f2fae9baa

• **Elfutils (0.192) – 11.635 KB:**
Página principal: https://sourceware.org/elfutils/
Descarga: https://sourceware.org/ftp/elfutils/0.192/elfutils-0.192.tar.bz2
Suma MD5: a6bb1efc147302cfc15b5c2b827f186a

• **Expat (2.6.4) - 476 KB:**
Página principal: https://libexpat.github.io/
Descarga: https://prdownloads.sourceforge.net/expat/expat-2.6.4.tar.xz
Suma MD5: 101fe3e320a2800f36af8cf4045b45c7

• **Expect (5.45.4) - 618 KB:**
Página principal: https://core.tcl.tk/expect/
Descarga: https://prdownloads.sourceforge.net/expect/expect5.45.4.tar.gz
Suma MD5: 00fce8de158422f5ccd2666512329bd2

• **File (5.46) - 1.283 KB:**
Página principal: https://www.darwinsys.com/file/
Descarga: https://astron.com/pub/file/file-5.46.tar.gz
Suma MD5: 459da2d4b534801e2e2861611d823864

• **Findutils (4.10.0) - 2.189 KB:**
Página principal: https://www.gnu.org/software/findutils/
Descarga: https://ftp.gnu.org/gnu/findutils/findutils-4.10.0.tar.xz
Suma MD5: 870cfd71c07d37ebe56f9f4aaf4ad872

• **Flex (2.6.4) - 1.386 KB:**
Página principal: https://github.com/westes/flex
Descarga: https://github.com/westes/flex/releases/download/v2.6.4/flex-2.6.4.tar.gz
Suma MD5: 2882e3179748cc9f9c23ec593d6adc8d

• **Flit-core (3.11.0) - 51 KB:**
Página principal: https://pypi.org/project/flit-core/
Descarga: https://pypi.org/packages/source/f/flit-core/flit_core-3.11.0.tar.gz
Suma MD5: 6d677b1acef1769c4c7156c7508e0dbd

• **Gawk (5.3.1) - 3.428 KB:**
Página principal: https://www.gnu.org/software/gawk/
Descargar: https://ftp.gnu.org/gnu/gawk/gawk-5.3.1.tar.xz
Suma MD5: 4e9292a06b43694500e0620851762eec

• **GCC (14.2.0) - 90.144 KB:**
Página principal: https://gcc.gnu.org/
Descargar: https://ftp.gnu.org/gnu/gcc/gcc-14.2.0/gcc-14.2.0.tar.xz
Suma MD5: 2268420ba02dc01821960e274711bde0

• **GDBM (1.24) - 1.168 KB:**
Página principal: https://www.gnu.org/software/gdbm/
Descargar: https://ftp.gnu.org/gnu/gdbm/gdbm-1.24.tar.gz
Suma MD5: c780815649e52317be48331c1773e987

• **Gettext (0.24) - 8.120 KB:**
Página principal: https://www.gnu.org/software/gettext/
Descargar: https://ftp.gnu.org/gnu/gettext/gettext-0.24.tar.xz
Suma MD5: 87aea3013802a3c60fa3feb5c7164069

• **Glibc (2.41) - 18.892 KB:**
Página principal: https://www.gnu.org/software/libc/
Descargar: https://ftp.gnu.org/gnu/glibc/glibc-2.41.tar.xz
Suma MD5: 19862601af60f73ac69e067d3e9267d4

Nota

Los desarrolladores de Glibc mantienen una rama de Git con parches considerados útiles para Glibc-2.41, pero que, lamentablemente, se desarrollaron después de su lanzamiento. Los editores de LFS emitirán un aviso de seguridad si se añade alguna corrección de seguridad a la rama, pero no se tomarán medidas para otros parches nuevos. Puede revisar los parches usted mismo e incorporar algunos si los considera importantes.

• **GMP (6.3.0) - 2.046 KB:**
Página principal: https://www.gnu.org/software/gmp/
Descarga: https://ftp.gnu.org/gnu/gmp/gmp-6.3.0.tar.xz
Suma MD5: 956dc04e864001a9c22429f761f2c283

• **Gperf (3.1) - 1.188 KB:**
Página principal: https://www.gnu.org/software/gperf/
Descarga: https://ftp.gnu.org/gnu/gperf/gperf-3.1.tar.gz
Suma MD5: 9e251c0a618ad0824b51117d5d9db87e

• **Grep (3.11) - 1.664 KB:**
Página principal: https://www.gnu.org/software/grep/
Descarga: https://ftp.gnu.org/gnu/grep/grep-3.11.tar.xz
Suma MD5: 7c9bbd74492131245f7cdb291fa142c0

• **Groff (1.23.0) - 7.259 KB:**
Página principal: https://www.gnu.org/software/groff/
Descarga: https://ftp.gnu.org/gnu/groff/groff-1.23.0.tar.gz
Suma MD5: 5e4f40315a22bb8a158748e7d5094c7d

• **GRUB (2.12) - 6.524 KB:**
Página principal: https://www.gnu.org/software/grub/
Descargar: https://ftp.gnu.org/gnu/grub/grub-2.12.tar.xz
Suma MD5: 60c564b1bdc39d8e43b3aab4bc0fb140

• **Gzip (1.13) - 819 KB:**
Página principal: https://www.gnu.org/software/gzip/
Descargar: https://ftp.gnu.org/gnu/gzip/gzip-1.13.tar.xz
Suma MD5: d5c9fc9441288817a4a0be2da0249e29

• **Iana-Etc (20250123) - 591 KB:**
Página principal: https://www.iana.org/protocols
Descargar: https://github.com/Mic92/iana-etc/releases/download/20250123/iana-etc-20250123.tar.gz
Suma MD5: f8a0ebdc19a5004cf42d8bdcf614fa5d

• **Inetutils (2.6) - 1.724 KB:**
Página principal: https://www.gnu.org/software/inetutils/
Descarga: https://ftp.gnu.org/gnu/inetutils/inetutils-2.6.tar.xz
Suma MD5: 401d7d07682a193960bcdecafd03de94

• **Intltool (0.51.0) - 159 KB:**
Página principal: https://freedesktop.org/wiki/Software/intltool
Descarga: https://launchpad.net/intltool/trunk/0.51.0/+download/intltool-0.51.0.tar.gz
Suma MD5: 12e517cac2b57a0121cda351570f1e63

• **IPRoute2 (6.13.0) - 906 KB:**
Página principal: https://www.kernel.org/pub/linux/utils/net/iproute2/
Descarga: https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-6.13.0.tar.xz
Suma MD5: 1603d25120d03feeaba9b360d03ffaec

• **Jinja2 (3.1.5) - 239 KB:**
Página principal: https://jinja.palletsprojects.com/en/3.1.x/
Descarga: https://pypi.org/packages/source/J/Jinja2/jinja2-3.1.5.tar.gz
Suma MD5: 083d64f070f6f1b5f75971ae60240785

• **Kbd (2.7.1) - 1.438 KB:**
Página principal: https://kbd-project.org/
Descarga: https://www.kernel.org/pub/linux/utils/kbd/kbd-2.7.1.tar.xz
Suma MD5: f15673d9f748e58f82fa50cff0d0fd20

• **Kmod (34) - 331 KB:**
Página principal: https://github.com/kmod-project/kmod
Descarga: https://www.kernel.org/pub/linux/utils/kernel/kmod/kmod-34.tar.xz
Suma MD5: 3e6c5c9ad9c7367ab9c3cc4f08dfde62

• **Less (668) - 635 KB:**
Página principal: https://www.greenwoodsoftware.com/less/
Descarga: https://www.greenwoodsoftware.com/less/less-668.tar.gz
Suma MD5: d72760386c5f80702890340d2f66c302

• **LFS-Bootscripts (20240825) - 33 KB:**
Descarga: https://www.linuxfromscratch.org/lfs/downloads/12.3/lfs-bootscripts-20240825.tar.xz
Suma MD5: 7b078c594a77e0f9cd53a0027471c3bc

• **Libcap (2.73) - 191 KB:**
Página principal: https://sites.google.com/site/fullycapable/
Descarga: https://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.73.tar.xz
Suma MD5: 0e186df9de9b1e925593a96684fe2e32

• **Libffi (3.4.7) - 1.362 KB:**
Página principal: https://sourceware.org/libffi/
Descarga: https://github.com/libffi/libffi/releases/download/v3.4.7/libffi-3.4.7.tar.gz
Suma MD5: 696a1d483a1174ce8a477575546a5284

• **Libpipeline (1.5.8) - 1.046 KB:**
Página principal: https://libpipeline.nongnu.org/
Descarga: https://download.savannah.gnu.org/releases/libpipeline/libpipeline-1.5.8.tar.gz
Suma MD5: 17ac6969b2015386bcb5d278a08a40b5

• **Libtool (2.5.4) - 1.033 KB:**
Página principal: https://www.gnu.org/software/libtool/
Descarga: https://ftp.gnu.org/gnu/libtool/libtool-2.5.4.tar.xz
Suma MD5: 22e0a29df8af5fdde276ea3a7d351d30

• **Libxcrypt (4.4.38) - 612 KB:**
Página principal: https://github.com/besser82/libxcrypt/
Descarga: https://github.com/besser82/libxcrypt/releases/download/v4.4.38/libxcrypt-4.4.38.tar.xz
Suma MD5: 1796a5d20098e9dd9e3f576803c83000

• **Linux (6.13.4) - 145.015 KB:**
Página principal: https://www.kernel.org/
Descarga: https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.13.4.tar.xz
Suma MD5: 13b9e6c29105a34db4647190a43d1810

Nota

El kernel de Linux se actualiza con frecuencia, muchas veces debido al descubrimiento de vulnerabilidades de seguridad. Se puede usar la última versión estable del kernel disponible, a menos que la página de erratas indique lo contrario.
Para usuarios con velocidad limitada o ancho de banda elevado que deseen actualizar el kernel de Linux, se puede descargar por separado una versión base del paquete y los parches. Esto puede ahorrar tiempo o dinero para una posterior actualización de parches dentro de una versión menor.

• **Lz4 (1.10.0) - 379 KB:**
Página principal: https://lz4.org/
Descarga: https://github.com/lz4/lz4/releases/download/v1.10.0/lz4-1.10.0.tar.gz
Suma MD5: dead9f5f1966d9ae56e1e32761e4e675

• **M4 (1.4.19) - 1.617 KB:**
Página principal: https://www.gnu.org/software/m4/
Descarga: https://ftp.gnu.org/gnu/m4/m4-1.4.19.tar.xz
Suma MD5: 0d90823e1426f1da2fd872df0311298d

• **Make (4.4.1) - 2.300 KB:**
Página principal: https://www.gnu.org/software/make/
Descarga: https://ftp.gnu.org/gnu/make/make-4.4.1.tar.gz
Suma MD5: c8469a3713cbbe04d955d4ae4be23eeb

• **Man-DB (2.13.0) - 2.023 KB:**
Página principal: https://www.nongnu.org/man-db/
Descarga: https://download.savannah.gnu.org/releases/man-db/man-db-2.13.0.tar.xz
Suma MD5: 97ab5f9f32914eef2062d867381d8cee

• **Man-Pages (6.12) - 1.838 KB:**
Página principal: https://www.kernel.org/doc/man-pages/
Descargar: https://www.kernel.org/pub/linux/docs/man-pages/man-pages-6.12.tar.xz
Suma MD5: 44de430a598605eaba3e36dd43f24298

• **MarkupSafe (3.0.2) - 21 KB:**
Página principal: https://palletsprojects.com/p/markupsafe/
Descargar: https://pypi.org/packages/source/M/MarkupSafe/markupsafe-3.0.2.tar.gz
Suma MD5: cb0071711b573b155cc8f86e1de72167

• **Meson (1.7.0) - 2.241 KB:**
Página principal: https://mesonbuild.com
Descargar: https://github.com/mesonbuild/meson/releases/download/1.7.0/meson-1.7.0.tar.gz
Suma MD5: c20f3e5ebbb007352d22f4fd6ceb925c

• **MPC (1.3.1) - 756 KB:**
Página principal: https://www.multiprecision.org/
Descarga: https://ftp.gnu.org/gnu/mpc/mpc-1.3.1.tar.gz
Suma MD5: 5c9bc658c9fd0f940e8e3e0f09530c62

• **MPFR (4.2.1) - 1.459 KB:**
Página principal: https://www.mpfr.org/
Descarga: https://ftp.gnu.org/gnu/mpfr/mpfr-4.2.1.tar.xz
Suma MD5: 523c50c6318dde6f9dc523bc0244690a

• **Ncurses (6.5) - 2.156 KB:**
Página principal: https://www.gnu.org/software/ncurses/
Descarga: https://invisible-mirror.net/archives/ncurses/ncurses-6.5.tar.gz
Suma MD5: ac2d2629296f04c8537ca706b6977687

• **Ninja (1.12.1) - 235 KB:**
Página principal: https://ninja-build.org/
Descarga: https://github.com/ninja-build/ninja/archive/v1.12.1/ninja-1.12.1.tar.gz
Suma MD5: 6288992b05e593a391599692e2f7e490

• **OpenSSL (3.4.1) – 17.917 KB:**
Página principal: https://www.openssl-library.org/
Descarga: https://github.com/openssl/openssl/releases/download/openssl-3.4.1/openssl-3.4.1.tar.gz
Suma MD5: fb7a747ac6793a7ad7118eaba45db379

• **Patch (2.7.6) - 766 KB:**
Página principal: https://savannah.gnu.org/projects/patch/
Descarga: https://ftp.gnu.org/gnu/patch/patch-2.7.6.tar.xz
Suma MD5: 78ad9937e4caadcba1526ef1853730d5

• **Perl (5.40.1) - 13.605 KB:**
Página principal: https://www.perl.org/
Descarga: https://www.cpan.org/src/5.0/perl-5.40.1.tar.xz
Suma MD5: bab3547a5cdf2302ee0396419d74a42e

• **Pkgconf (2.3.0) - 309 KB:**
Página principal: https://github.com/pkgconf/pkgconf
Descarga: https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz
Suma MD5: 833363e77b5bed0131c7bc4cc6f7747b

• **Procps (4.0.5) - 1.483 KB:**
Página principal: https://gitlab.com/procps-ng/procps/
Descarga: https://sourceforge.net/projects/procps-ng/files/Production/procps-ng-4.0.5.tar.xz
Suma MD5: 90803e64f51f192f3325d25c3335d057

• **Psmisc (23.7) - 423 KB:**
Página principal: https://gitlab.com/psmisc/psmisc
Descarga: https://sourceforge.net/projects/psmisc/files/psmisc/psmisc-23.7.tar.xz
Suma MD5: 53eae841735189a896d614cba440eb10

• **Python (3.13.2) - 22.091 KB:**
Página principal: https://www.python.org/
Descarga: https://www.python.org/ftp/python/3.13.2/Python-3.13.2.tar.xz
Suma MD5: 4c2d9202ab4db02c9d0999b14655dfe5

• **Python Documentation (3.13.2) - 10.102 KB:**
Descarga: https://www.python.org/ftp/python/doc/3.13.2/python-3.13.2-docs-html.tar.bz2
Suma MD5: d6aede88f480a018d26b3206f21654ae

• **Readline (8.2.13) - 29.74 KB:**
Página principal: https://tiswww.case.edu/php/chet/readline/rltop.html
Descarga: https://ftp.gnu.org/gnu/readline/readline-8.2.13.tar.gz
Suma MD5: 05080bf3801e6874bb115cd6700b708f

• **Sed (4.9) - 1.365 KB:**
Página principal: https://www.gnu.org/software/sed/
Descarga: https://ftp.gnu.org/gnu/sed/sed-4.9.tar.xz
Suma MD5: 6aac9b2dbafcd5b7a67a8a9bcb8036c3

• **Setuptools (75.8.1) - 1.313 KB:**
Página principal: https://pypi.org/project/setuptools/
Descarga: https://pypi.org/packages/source/s/setuptools/setuptools-75.8.1.tar.gz
Suma MD5: 7dc3d3f529b76b10e35326e25c676b30

• **Shadow (4.17.3) - 2.274 KB:**
Página principal: https://github.com/shadow-maint/shadow/
Descarga: https://github.com/shadow-maint/shadow/releases/download/4.17.3/shadow-4.17.3.tar.xz
Suma MD5: 0da190e53ecee76237e4c8f3f39531ed

• **Sysklogd (2.7.0) - 465 KB:**
Página principal: https://www.infodrom.org/projects/sysklogd/
Descarga: https://github.com/troglobit/sysklogd/releases/download/v2.7.0/sysklogd-2.7.0.tar.gz
Suma MD5: 611c0fa5c138eb7a532f3c13bdf11ebc

• **Systemd (257.3) - 15.847 KB:**
Página principal: Español: https://www.freedesktop.org/wiki/Software/systemd/
Descargar: https://github.com/systemd/systemd/archive/v257.3/systemd-257.3.tar.gz
Suma MD5: 8e4fc90c7aead651fa5c50bd1b34abc2

• **Systemd Man Pages (257.3) - 733 KB:**
Página principal: https://www.freedesktop.org/wiki/Software/systemd/
Descargar: https://anduin.linuxfromscratch.org/LFS/systemd-man-pages-257.3.tar.xz
Suma MD5: 9b77c3b066723d490cb10aed4fb05696

Nota

El equipo de Linux From Scratch genera su propio archivo tar de las páginas de manual utilizando el código fuente de systemd. Esto se hace para evitar dependencias innecesarias.

• **SysVinit (3.14) - 236 KB:**
Página principal: https://savannah.nongnu.org/projects/sysvinit
Descarga: https://github.com/slicer69/sysvinit/releases/download/3.14/sysvinit-3.14.tar.xz
Suma MD5: bc6890b975d19dc9db42d0c7364dd092

• **Tar (1.35) - 2.263 KB:**
Página principal: https://www.gnu.org/software/tar/
Descarga: https://ftp.gnu.org/gnu/tar/tar-1.35.tar.xz
Suma MD5: a2d8042658cfd8ea939e6d911eaf4152

• **Tcl (8.6.16) - 11.406 KB:**
Página principal: https://tcl.sourceforge.net/
Descarga: https://downloads.sourceforge.net/tcl/tcl8.6.16-src.tar.gz
Suma MD5: eaef5d0a27239fb840f04af8ec608242

• **Tcl Documentation (8.6.16) - 1.169 KB:**
Descarga: https://downloads.sourceforge.net/tcl/tcl8.6.16-html.tar.gz
Suma MD5: 750c221bcb6f8737a6791c1fbe98b684

• **Texinfo (7.2) - 6.259 KB:**
Página principal: https://www.gnu.org/software/texinfo/
Descargar: https://ftp.gnu.org/gnu/texinfo/texinfo-7.2.tar.xz
Suma MD5: 11939a7624572814912a18e76c8d8972

• **Time Zone Data (2025a) - 453 KB:**
Página principal: https://www.iana.org/time-zones
Descargar: https://www.iana.org/time-zones/repository/releases/tzdata2025a.tar.gz
Suma MD5: 404229390c06b7440f5e48d12c1a3251

• **Udev-lfs Tarball (udev-lfs-20230818) - 10 KB:**
Descargar: https://anduin.linuxfromscratch.org/LFS/udev-lfs-20230818.tar.xz
Suma MD5: acd4360d8a5c3ef320b9db88d275dae6

• **Util-linux (2.40.4) - 8.641 KB:**
Página principal: https://git.kernel.org/pub/scm/utils/util-linux/util-linux.git/
Descarga: https://www.kernel.org/pub/linux/utils/util-linux/v2.40/util-linux-2.40.4.tar.xz
Suma MD5: f9cbb1c8315d8ccbeb0ec36d10350304

• **Vim (9.1.1166) - 18.077 KB:**
Página principal: https://www.vim.org
Descarga: https://github.com/vim/vim/archive/v9.1.1166/vim-9.1.1166.tar.gz
Suma MD5: 718d43ce957ab7c81071793de176c2eb

Nota

La versión de Vim cambia a diario. Para obtener la versión más reciente, visite https://github.com/vim/vim/tags.

• **Wheel (0.45.1) - 106 KB:**
Página principal: https://pypi.org/project/wheel/
Descarga: https://pypi.org/packages/source/w/wheel/wheel-0.45.1.tar.gz
Suma MD5: dddc505d0573d03576c7c6c5a4fe0641

• **XML::Parser (2.47) - 276 KB:**
Página principal: https://github.com/chorny/XML-Parser
Descarga: https://cpan.metacpan.org/authors/id/T/TO/TODDR/XML-Parser-2.47.tar.gz
Suma MD5: 89a8e82cfd2ad948b349c0a69c494463

• **Xz Utils (5.6.4) - 1.310 KB:**
Página principal: https://tukaani.org/xz
Descarga: https://github.com//tukaani-project/xz/releases/download/v5.6.4/xz-5.6.4.tar.xz
Suma MD5: 4b1cf07d45ec7eb90a01dd3c00311a3e

• **Zlib (1.3.1) - 1.478 KB:**
Página principal: https://zlib.net/
Descarga: https://zlib.net/fossils/zlib-1.3.1.tar.gz
Suma MD5: 9855b6d802d7fe5b7bd5b196a2271655

• **Zstd (1.5.7) - 2.378 KB:**
Página principal: https://facebook.github.io/zstd/
Descarga: https://github.com/facebook/zstd/releases/download/v1.5.7/zstd-1.5.7.tar.gz
Suma MD5: 780fc1896922b1bc52a4e90980cdda48

Tamaño total de estos paquetes: **Aproximadamente 527 MB**

3.3. Parches necesarios

Además de los paquetes, se requieren varios parches. Estos parches corrigen errores en los paquetes que el mantenedor debería corregir. También realizan pequeñas modificaciones para facilitar el uso de los paquetes. Se necesitarán los siguientes parches para compilar un sistema LFS:

• Bzip2 Documentation Patch - 1.6 KB:
Descargar: https://www.linuxfromscratch.org/patches/lfs/12.3/bzip2-1.0.8-install_docs-1.patch
Suma MD5: 6a5ac7e89b791aae556de0f745916f7f

• Coreutils Internationalization Fixes Patch - 164 KB:
Descargar: https://www.linuxfromscratch.org/patches/lfs/12.3/coreutils-9.6-i18n-1.patch
Suma MD5: 6aee45dd3e05b7658971c321d92f44b7

• Expect GCC14 Patch - 7.8 KB:
Descargar: https://www.linuxfromscratch.org/patches/lfs/12.3/expect-5.45.4-gcc14-1.patch
Suma MD5: 0b8b5ac411d011263ad40b0664c669f0

• Glibc FHS Patch - 2.8 KB:
Descargar: https://www.linuxfromscratch.org/patches/lfs/12.3/glibc-2.41-fhs-1.patch
Suma MD5: 9a5997c3452909b1769918c759eff8a2

• Kbd Backspace/Delete Fix Patch - 12 KB:
Descargar: https://www.linuxfromscratch.org/patches/lfs/12.3/kbd-2.7.1-backspace-1.patch
MD5 Suma: f75cca16a38da6caa7d52151f7136895

• SysVinit Consolidated Patch- 2.5 KB:
Descarga: https://www.linuxfromscratch.org/patches/lfs/12.3/sysvinit-3.14-consolidated-1.patch
Suma MD5: 3af8fd8e13cad481eeeaa48be4247445

Tamaño total de estos parches: **Aproximadamente 190.7 KB**

Además de los parches obligatorios mencionados, existen varios parches opcionales creados por la comunidad LFS. Estos parches opcionales solucionan problemas menores o habilitan funciones no habilitadas por defecto. Puede consultar la base de datos de parches en *https://www.linuxfromscratch.org/patches/downloads/* y adquirir parches adicionales que se ajusten a las necesidades de su sistema.

Capítulo 4. Preparativos Finales

4.1. Introducción

En este capítulo, realizaremos algunas tareas adicionales para preparar la compilación del sistema temporal. Crearemos un conjunto de directorios en $LFS (donde instalaremos las herramientas temporales), agregaremos un usuario sin privilegios y crearemos un entorno de compilación adecuado para dicho usuario. También explicaremos las unidades de tiempo (SBU) que utilizamos para medir el tiempo de compilación de paquetes LFS y proporcionaremos información sobre los conjuntos de pruebas de paquetes.

4.2. Creación de una distribución de directorios limitada en el sistema de archivos LFS

En esta sección, comenzamos a poblar el sistema de archivos LFS con los componentes que conformarán el sistema Linux final.

El primer paso es crear una jerarquía de directorios limitada para que los programas compilados en el Capítulo 6 (así como glibc y libstdc++ en el Capítulo 5) puedan instalarse en su ubicación final. Hacemos esto para que esos programas temporales se sobrescriban al compilar las versiones finales en el Capítulo 8.

Cree la distribución de directorios necesaria ejecutando los siguientes comandos como root:

```
mkdir -pv $LFS/{etc,var} $LFS/usr/{bin,lib,sbin}

for i in bin lib sbin; do
  ln -sv usr/$i $LFS/$i
done

case $(uname -m) in
  x86_64) mkdir -pv $LFS/lib64 ;;
esac
```

Los programas del Capítulo 6 se compilarán con un compilador cruzado (más detalles en la sección "Notas técnicas de la cadena de herramientas"). Este compilador cruzado se instalará en un directorio especial para separarlo de los demás programas. Aún como root, cree ese directorio con este comando:

```
mkdir -pv $LFS/tools
```

Nota

Los editores de LFS han decidido deliberadamente no usar el directorio /usr/lib64. Se toman varias medidas para garantizar que las herramientas no lo usen. Si por alguna razón aparece este directorio (ya sea por un error al seguir las instrucciones o porque instaló un paquete binario que lo creó después de finalizar LFS), podría dañar su sistema. Asegúrese siempre de que este directorio no exista.

4.3. Añadiendo el usuario de LFS

Al iniciar sesión como root, un solo error puede dañar o destruir el sistema. Por lo tanto, los paquetes de los dos siguientes capítulos se compilan con un usuario sin privilegios. Puede usar su propio nombre de usuario, pero para facilitar la configuración de un entorno de trabajo limpio, crearemos un nuevo usuario llamado lfs como miembro de un nuevo grupo (también llamado lfs) y ejecutaremos comandos como lfs durante el proceso de instalación. Como root, ejecute los siguientes comandos para agregar el nuevo usuario:

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

Significado las opciones de la línea de comandos:

`-s /bin/bash`
: Esto convierte a **bash** en el shell predeterminado para el usuario lfs.

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

-m
: Esto crea un directorio personal para lfs.

`-k /dev/null`
: Este parámetro evita la posible copia de archivos desde un directorio esqueleto (el predeterminado es /etc/skel) al cambiar la ubicación de entrada al dispositivo nulo especial.

`lfs`
: Este es el nombre del nuevo usuario.

Si desea iniciar sesión como lfs o cambiar a lfs desde un usuario no root (a diferencia de cambiar al usuario lfs cuando se inicia sesión como root, lo cual no requiere que el usuario lfs tenga una contraseña), debe establecer una contraseña para lfs. Ejecute el siguiente comando como usuario root para establecer la contraseña:

```
passwd lfs
```

Conceda a lfs acceso completo a todos los directorios bajo $LFS, convirtiéndolo en el propietario:

```
chown -v lfs $LFS/{usr{,/*},var,etc,tools}
case $(uname -m) in
  x86_64) chown -v lfs $LFS/lib64 ;;
esac
```

Nota
En algunos sistemas host, el siguiente comando su no se completa correctamente y suspende el inicio de sesión del usuario lfs en segundo plano. Si el indicador "lfs:~$" no aparece inmediatamente, el problema se solucionará con el comando fg.

A continuación, inicie un shell con el usuario lfs. Esto se puede hacer iniciando sesión como lfs en una consola virtual o con el siguiente comando de sustitución/cambio de usuario:

```
su - lfs
```

El símbolo "-" indica a su que inicie un shell con inicio de sesión en lugar de uno sin inicio de sesión. La diferencia entre estos dos tipos de shells se describe en detalle en *bash(1)* e info bash.

4.4. Configuración del entorno

Configure un buen entorno de trabajo creando dos nuevos archivos de inicio para el shell **bash**. Con la sesión iniciada como usuario lfs, ejecute el siguiente comando para crear un nuevo archivo .bash_profile:

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

Al iniciar sesión como usuario lfs, o al cambiar al usuario lfs mediante el comando su con la opción "-", el shell inicial es un shell de inicio de sesión que lee el archivo /etc/profile del host (que probablemente contiene algunas configuraciones y variables de entorno) y luego el archivo .bash_profile. El comando **exec env -i.../bin/bash** del archivo .bash_profile reemplaza el shell en ejecución por uno nuevo con un entorno completamente vacío, excepto por las variables HOME, TERM y PS1. Esto garantiza que ninguna variable de entorno no deseada y potencialmente peligrosa del sistema host se filtre al entorno de compilación.

La nueva instancia del shell es una shell sin inicio de sesión, que no lee ni ejecuta el contenido de los archivos /etc/profile ni .bash_profile, sino que lee y ejecuta el archivo .bashrc. Cree el archivo .bashrc ahora:

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
LFS=/mnt/lfs
LC_ALL=POSIX
LFS_TGT=$(uname -m)-lfs-linux-gnu
PATH=/usr/bin
if [ ! -L /bin ]; then PATH=/bin:$PATH; fi
PATH=$LFS/tools/bin:$PATH
CONFIG_SITE=$LFS/usr/share/config.site
export LFS LC_ALL LFS_TGT PATH CONFIG_SITE
EOF
```

Significado de la configuración en .bashrc:

set +h

El comando **set +h** desactiva la función hash de **bash**. El hash suele ser una función útil: **bash** utiliza una tabla hash para recordar la ruta completa de los archivos ejecutables y evitar buscar repetidamente en PATH el mismo ejecutable. Sin embargo, las nuevas herramientas deben usarse tan pronto como se instalen. Desactivar la función hash obliga al shell a buscar en PATH cada vez que se ejecuta un programa. Por lo tanto, el shell encontrará las herramientas recién compiladas en $LFS/tools/bin tan pronto como estén disponibles, sin recordar una versión anterior del mismo programa proporcionada por la distribución del host, en /usr/bin o /bin.

umask 022

Configurando la máscara de usuario como ya explicamos en la Sección 2.6, “Configuración de la variable $LFS y la máscara de usuario”.

LFS=/mnt/lfs

La variable LFS debe establecerse en el punto de montaje seleccionado.

LC_ALL=POSIX

La variable LC_ALL controla la localización de ciertos programas, haciendo que sus mensajes sigan las convenciones de un país específico. Establecer LC_ALL en “POSIX” o “C” (ambos son equivalentes) garantiza que todo funcione correctamente en el entorno de compilación cruzada.

LFS_TGT=$(uname -m)-lfs-linux-gnu

La variable LFS_TGT establece una descripción de máquina no predeterminada, pero compatible, para usarla al compilar nuestro compilador y enlazador cruzados, así como al compilar nuestra cadena de herramientas temporal. Para más información, consulte las Notas Técnicas de la Cadena de Herramientas.

PATH=/usr/bin

Muchas distribuciones modernas de Linux han fusionado /bin y /usr/bin. En este caso, la variable PATH estándar debe establecerse en /usr/bin/ para el entorno del Capítulo 6. En caso contrario, la siguiente línea añade /bin a la ruta.

if [! -L /bin]; then PATH=/bin:$PATH; fi

Si /bin no es un enlace simbólico, debe añadirse a la variable PATH.

PATH=$LFS/tools/bin:$PATH

Al colocar $LFS/tools/bin antes de la variable PATH estándar, el compilador cruzado instalado al principio del Capítulo 5 es recogido por el shell inmediatamente después de su instalación. Esto, junto con la desactivación del hash, limita el riesgo de que se utilice el compilador del host en lugar del compilador cruzado.

CONFIG_SITE=$LFS/usr/share/config.site

En los capítulos 5 y 6, si esta variable no está configurada, los scripts de configuración podrían intentar cargar elementos de configuración específicos de algunas distribuciones desde /usr/share/config.site en el sistema host. Anótela para evitar una posible contaminación desde el host.

export ...

Aunque los comandos anteriores han configurado algunas variables, para que sean visibles en cualquier subshell, las exportamos.

Importante

Varias distribuciones comerciales añaden una instancia no documentada de /etc/bash.bashrc a la inicialización de bash. Este archivo puede modificar el entorno del usuario lfs de forma que pueda afectar la compilación de paquetes LFS críticos. Para asegurarse de que el entorno del usuario lfs esté limpio, compruebe la presencia de /etc/bash.bashrc y, si está presente, elimínelo. Como usuario root, ejecute:

```
[ ! -e /etc/bash.bashrc ] || mv -v /etc/bash.bashrc /etc/bash.bashrc.NOUSE
```

Cuando el usuario lfs ya no sea necesario (al principio del Capítulo 7), puede restaurar /etc/bash.
bashrc de forma segura (si lo desea).
Tenga en cuenta que el paquete LFS Bash que crearemos en la Sección 8.36, “Bash-5.2.37” no está configurado para cargar ni ejecutar /etc/bash.bashrc, por lo que este archivo es inútil en un sistema LFS completo.

En muchos sistemas modernos con varios procesadores (o núcleos), el tiempo de compilación de un paquete se puede reducir realizando una "compilación paralela" indicando al programa cuántos procesadores están disponibles mediante una opción de línea de comandos o una variable de entorno. Por ejemplo, un procesador Intel Core i9-13900K tiene 8 núcleos P (de rendimiento) y 16 núcleos E (de eficiencia). Un núcleo P puede ejecutar dos subprocesos simultáneamente, por lo que el kernel de Linux modela cada núcleo P como dos núcleos lógicos. Como resultado, hay 32 núcleos lógicos en total. Una forma obvia de usar todos estos núcleos lógicos es permitir que **make** genere hasta 32 trabajos de compilación. Esto se puede hacer pasando la opción -j32 a **make**:

```
make -j32
```

O bien, configurando la variable de entorno MAKEFLAGS, cuyo contenido será utilizado automáticamente por **make** como opción de línea de comandos:

```
export MAKEFLAGS=-j32
```

Importante

Nunca pase una opción -j sin un número para **make** ni configure dicha opción en MAKEFLAGS. Hacerlo permitiría que **make** genere trabajos de compilación infinitos y causara problemas de estabilidad del sistema.

Para usar todos los núcleos lógicos disponibles para compilar paquetes en los capítulos 5 y 6, configure MAKEFLAGS ahora en .bashrc:

```
cat >> ~/.bashrc << "EOF"
export MAKEFLAGS=-j$(nproc)
EOF
```

Reemplace $(nproc) con el número de núcleos lógicos que desea usar si no desea usarlos todos.

Finalmente, para asegurar que el entorno esté completamente preparado para compilar las herramientas temporales, fuerce la shell bash a leer el nuevo perfil de usuario:

```
source ~/.bash_profile
```

4.5. Acerca de las Standard Build Unit o SBU

Muchas personas desean saber de antemano cuánto tiempo se tarda aproximadamente en compilar e instalar cada paquete.

Dado que Linux From Scratch puede compilarse en muchos sistemas diferentes, es imposible proporcionar estimaciones de tiempo absolutas.

El paquete más grande (gcc) tardará aproximadamente 5 minutos en los sistemas más rápidos, pero podría tardar días en sistemas más lentos. En lugar de proporcionar tiempos reales, se utilizará la unidad de compilación estándar (SBU).

La unidad de compilación estándar (SBU) funciona de la siguiente manera: el primer paquete que se compilará es binutils en el Capítulo 5. El tiempo que tarda la compilación con un núcleo se denominará Unidad de Compilación Estándar o SBU. Todos los demás tiempos de compilación se expresarán en esta unidad de tiempo.

Por ejemplo, considere un paquete cuyo tiempo de compilación es de 4,5 SBU. Esto significa que si su sistema tardó 4 minutos en compilar e instalar la primera pasada de binutils, tardará *aproximadamente* 18 minutos en compilar el paquete de ejemplo. Afortunadamente, la mayoría de los tiempos de compilación son inferiores a una SBU.

Las SBU no son del todo precisas, ya que dependen de muchos factores, incluida la versión de GCC del sistema host. Se proporcionan aquí para ofrecer una estimación de cuánto tiempo podría tardar la instalación de un paquete, pero las cifras pueden variar hasta en decenas de minutos en algunos casos.

En algunos sistemas más recientes, la placa base puede controlar la velocidad del reloj del sistema. Esto se puede controlar con un comando como **powerprofilesctl**. Esto no está disponible en LFS, pero podría estarlo en la distribución host. Una vez completado LFS, se puede añadir al sistema mediante los procedimientos de la página power-profiles-daemon de BLFS. Antes de medir el tiempo de compilación de cualquier paquete, es recomendable utilizar un perfil de energía del sistema configurado para obtener el máximo rendimiento (y el máximo consumo de energía). De lo contrario, el valor de SBU medido puede ser inexacto, ya que el sistema puede reaccionar de forma diferente al compilar binutils-pass1 u otros paquetes. Tenga en cuenta que puede aparecer una inexactitud significativa incluso si se utiliza el mismo perfil para ambos paquetes, ya que el sistema puede responder más lento si está inactivo al iniciar el proceso de compilación. Configurar el perfil de energía en "rendimiento" minimizará este problema. Y, obviamente, al hacerlo, también acelerará la compilación de LFS del sistema.

Si **powerprofilesctl** está disponible, ejecute el comando **powerprofilesctl set performance** para seleccionar el perfil de rendimiento. Algunas distribuciones proporcionan el comando **tuned-adm** para administrar los perfiles en lugar de **powerprofilesctl**. En estas distribuciones, ejecute el comando **tuned-adm profile throughput-performance** para seleccionar el perfil de rendimiento (throughput-performance).

Nota

Cuando se utilizan varios procesadores de esta manera, las unidades SBU del libro variarán aún más de lo normal. En algunos casos, el paso de creación simplemente fallará. Analizar la salida del proceso de compilación también será más difícil porque las líneas de los diferentes procesos estarán intercaladas. Si surge un problema con un paso de compilación, vuelva a la compilación con un solo procesador para analizar correctamente los mensajes de error.
Los tiempos que se presentan aquí para todos los paquetes (excepto binutils-pass1, que se basa en un núcleo) se basan en el uso de cuatro núcleos (-j4). Los tiempos del Capítulo 8 también incluyen el tiempo necesario para ejecutar las pruebas de regresión del paquete, a menos que se especifique lo contrario.

4.6. Acerca de los conjuntos de pruebas

La mayoría de los paquetes proporcionan un conjunto de pruebas. Ejecutar el conjunto de pruebas para un paquete recién compilado es recomendable, ya que puede proporcionar una comprobación de integridad que indica que todo se compiló correctamente. Un conjunto de pruebas que supera sus comprobaciones suele demostrar que el paquete funciona según lo previsto por el desarrollador. Sin embargo, no garantiza que el paquete esté totalmente libre de errores.

Algunas suites de pruebas son más importantes que otras. Por ejemplo, las suites de pruebas para los paquetes principales de la cadena de herramientas (GCC, binutils y glibc) son cruciales debido a su papel fundamental en el correcto funcionamiento del sistema. Las suites de pruebas para GCC y glibc pueden tardar mucho tiempo en completarse, especialmente en hardware lento, pero se recomiendan encarecidamente.

Nota

Ejecutar las suites de pruebas de los capítulos 5 y 6 no tiene sentido; dado que los programas de prueba se compilan con un compilador cruzado, probablemente no puedan ejecutarse en el host de compilación.

Un problema común al ejecutar las suites de pruebas para binutils y GCC es quedarse sin pseudoterminales (PTY). Esto puede provocar un gran número de pruebas fallidas. Esto puede ocurrir por varias razones, pero la causa más probable es que el sistema host no tenga configurado correctamente el sistema de archivos devpts. Este problema se describe con más detalle en *https://www.linuxfromscratch.org/lfs/faq.html#no-ptys*.

En ocasiones, las suites de pruebas de paquetes fallan por razones que los desarrolladores conocen y que no consideran críticas.

Consulte los registros en *https://www.linuxfromscratch.org/lfs/build-logs/12.3/* para verificar si estos fallos son previsibles. Este sitio es válido para todas las suites de pruebas de este libro.

Parte III. Creación de la cadena de herramientas cruzadas de LFS y herramientas temporales

Material preliminar importante

Introducción

Esta parte se divide en tres etapas: primero, construir un compilador cruzado y sus bibliotecas asociadas; segundo, usar esta cadena de herramientas cruzada para construir varias utilidades de forma que se aíslen de la distribución del host; y tercero, entrar en el entorno chroot (que mejora aún más el aislamiento del host) y construir las herramientas restantes necesarias para construir el sistema final.

Importante

Aquí es donde comienza el verdadero trabajo de construir un nuevo sistema. Asegúrese de seguir las instrucciones exactamente como se muestran en el libro. Debe intentar comprender la función de cada comando y, por mucho que desee terminar la compilación, evite escribir los comandos a ciegas como se muestra. Lea la documentación cuando no entienda algo. Además, controle lo que escribe y la salida de los comandos, utilizando la utilidad tee para enviar la salida de la terminal a un archivo. Esto facilita la depuración si algo sale mal.

La siguiente sección es una introducción técnica al proceso de compilación, mientras que la siguiente presenta instrucciones generales muy importantes.

Notas técnicas de la cadena de compilación o Toolchain

Esta sección explica algunos de los fundamentos y detalles técnicos del método de compilación general. No intente comprender todo de inmediato en esta sección. La mayor parte de esta información quedará más clara después de realizar una compilación real. Vuelva a leer este capítulo en cualquier momento durante el proceso de compilación.

El objetivo general de los capítulos 5 y 6 es crear un área temporal que contenga un conjunto de herramientas de eficacia comprobada, aisladas del sistema host. Mediante el comando **chroot**, las compilaciones de los capítulos restantes se aislarán dentro de ese entorno, lo que garantiza una compilación limpia y sin problemas del sistema LFS de destino. El proceso de compilación se ha diseñado para minimizar los riesgos para los nuevos lectores y, al mismo tiempo, para ofrecer el máximo valor educativo.

Este proceso de compilación se basa en la compilación cruzada. La compilación cruzada se utiliza normalmente para compilar un compilador y su cadena de herramientas asociada para una máquina diferente a la utilizada para la compilación. Esto no es estrictamente necesario para LFS, ya que la máquina donde se ejecutará el nuevo sistema es la misma que se utilizó para la compilación. Sin embargo, la compilación cruzada tiene una gran ventaja: todo lo compilado de forma cruzada no puede depender del entorno del host.

Acerca de la compilación-cruzada

Nota

El libro de LFS no es (ni contiene) un tutorial general para construir una cadena de herramientas cruzada (o nativa). No utilice los comandos del libro para una cadena de herramientas cruzada con ningún otro propósito que no sea la compilación de LFS, a menos que realmente comprenda lo que está haciendo.

La compilación cruzada implica algunos conceptos que merecen una sección aparte. Aunque esta sección puede omitirse en una primera lectura, volver a ella más adelante le ayudará a comprender mejor el proceso.

Primero, definamos algunos términos utilizados en este contexto.

La compilación

es la máquina donde compilamos los programas. Tenga en cuenta que esta máquina también se denomina "host".

El host

es la máquina/sistema donde se ejecutarán los programas compilados. Tenga en cuenta que este uso de "host" no es el mismo que en otras secciones.

El objetivo

solo se usa para compiladores. Es la máquina para la que el compilador produce código. Puede ser diferente tanto de la compilación como del host.

Como ejemplo, imaginemos el siguiente escenario (a veces denominado "Cruce Canadiense"). Tenemos un compilador

solo en una máquina lenta, llamémosla máquina A, y el compilador ccA. También tenemos una máquina rápida (B), pero no hay compilador para (B), y queremos producir código para una tercera máquina lenta (C). Construiremos un compilador para la máquina C en tres etapas.

Etapa	Compilación	Host	Objetivo	Acción
1	A	A	B	Construir el compilador cruzado cc1 usando ccA en la máquina A.
2	A	B	C	Construir el compilador cruzado cc2 usando cc1 en la máquina A.
3	B	C	C	Construir el compilador cruzado ccC usando cc2 en la máquina B.

Entonces, todos los programas que necesita la máquina C se pueden compilar usando cc2 en la máquina rápida B. Tenga en cuenta que, a menos que B pueda ejecutar programas producidos para C, no hay forma de probar los programas recién compilados hasta que la máquina C esté ejecutándose. Por ejemplo, para ejecutar un conjunto de pruebas en ccC, podríamos añadir una cuarta etapa:

Etapa	Compilación	Host	Objetivo	Acción
4	C	C	C	Reconstruir y probar ccC usando ccC en la máquina C.

En el ejemplo anterior, solo cc1 y cc2 son compiladores cruzados; es decir, producen código para una máquina diferente de aquella en la que se ejecutan. Los otros compiladores, ccA y ccC, producen código para la máquina en la que se ejecutan. Estos compiladores se denominan compiladores nativos.

Implementación de la compilación cruzada para LFS

Nota

Todos los paquetes compilados cruzados de este libro utilizan un sistema de compilación basado en autoconf. Este sistema acepta tipos de sistema con la forma cpu-vendor-kernel-os, conocido como el triplete del sistema. Dado que el campo del proveedor suele ser irrelevante, autoconf permite omitirlo.
Un lector astuto podría preguntarse por qué un "triplete" se refiere a un nombre de cuatro componentes. Los campos kernel y os comenzaron como un único campo "sistema". Esta forma de tres campos sigue siendo válida hoy en día para algunos sistemas, por ejemplo, x86_64-unknown-freebsd. Pero dos sistemas pueden compartir el mismo kernel y aun así ser demasiado diferentes como para usar el mismo triplete para describirlos. Por ejemplo, Android ejecutándose en un teléfono móvil es completamente diferente de Ubuntu ejecutándose en un servidor ARM64, aunque ambos se ejecuten en el mismo tipo de CPU (ARM64) y usen el mismo kernel (Linux).
Sin una capa de emulación, no se puede ejecutar un ejecutable para un servidor en un teléfono móvil o viceversa. Por lo tanto, el campo "sistema" se ha dividido en campos de kernel y sistema operativo para designar estos sistemas inequívocamente. En nuestro ejemplo, el sistema Android se designa como aarch64-unknown-linux-android, y el sistema Ubuntu como aarch64-unknown-linux-gnu.
La palabra "triplete" permanece incrustada en el léxico. Una forma sencilla de determinar el triplete de su sistema es ejecutar el script **config.guess** que viene con el código fuente de muchos paquetes. Descomprima el código fuente de binutils, ejecute el script **./config.guess** y observe el resultado. Por ejemplo, para un procesador Intel de 32 bits, el resultado será i686-pc-linux-gnu. En un sistema de 64 bits, será x86_64-pc-linux-gnu. En la mayoría de los sistemas Linux, el comando **gcc-dumpmachine**, aún más sencillo, proporcionará información similar.
También debe tener en cuenta el nombre del enlazador dinámico de la plataforma, a menudo denominado cargador dinámico (no confundir con el enlazador estándar **ld**, que forma parte de binutils). El enlazador dinámico proporcionado por el paquete glibc encuentra y carga las bibliotecas compartidas que necesita un programa, lo prepara para su ejecución y luego lo ejecuta. El nombre del enlazador dinámico para un equipo Intel de 32 bits es ld-linux.so.2; en su lugar, es ld-linux-x86-64.so.2 en sistemas de 64 bits. Una forma segura de determinar el nombre del enlazador dinámico es inspeccionar un binario aleatorio del sistema host ejecutando: **readelf -l <nombre del binario> | grep interpreter** y anotando la salida. La referencia autorizada que cubre todas las plataformas se encuentra en una página wiki de Glibc.

Para simular una compilación cruzada en LFS, el nombre del triplete del host se ajusta ligeramente cambiando el campo "vendor" en la variable LFS_TGT para que diga "lfs". También usamos la opción *--with-sysroot* al compilar el enlazador y el compilador cruzados para indicarles dónde encontrar los archivos del host necesarios. Esto garantiza que ninguno de los otros programas compilados en el Capítulo 6 pueda enlazar con las bibliotecas en el equipo de compilación. Solo son obligatorias dos etapas, más una adicional para pruebas.

Etapa	Compilación	Host	Objetivo	Acción
1	pc	pc	lfs	Construir el compilador cruzado cc1 usando cc-pc en pc.
2	pc	lfs	lfs	Construir el compilador cruzado cc-lfs usando cc-1 en pc.
3	lfs	lfs	lfs	Reconstruir y probar cc-lfs usando cc-lfs en lfs.

En la tabla anterior, "en PC" significa que los comandos se ejecutan en una máquina con la distribución ya instalada. "En LFS" significa que los comandos se ejecutan en un entorno chroot.

Esto no es todo. El lenguaje C no es simplemente un compilador; también define una biblioteca estándar. En este libro, se utiliza la biblioteca GNU C, llamada glibc (existe una alternativa, "musl"). Esta biblioteca debe compilarse para la máquina LFS; es decir, utilizando el compilador cruzado cc1. Sin embargo, el compilador utiliza una biblioteca interna que proporciona subrutinas complejas para funciones no disponibles en el conjunto de instrucciones del ensamblador. Esta biblioteca interna se llama libgcc y debe estar enlazada a la biblioteca glibc para ser completamente funcional. Además, la biblioteca estándar para C++ (libstdc++) también debe estar enlazada con glibc. La solución a este dilema del huevo y la gallina es, primero, compilar una biblioteca libgcc degradada basada en cc1, que carece de algunas funcionalidades como hilos y gestión de excepciones. Después, compilar glibc utilizando este compilador degradado (glibc en sí no está degradado), y también compilar libstdc++. Esta última biblioteca carecerá de algunas de las funcionalidades de libgcc.

La consecuencia del párrafo anterior es que cc1 no puede compilar una biblioteca libstdc++ completamente funcional con la biblioteca libgcc degradada, pero cc1 es el único compilador disponible para compilar las bibliotecas de C/C++ durante la etapa 2. Hay dos razones por las que no usamos inmediatamente el compilador integrado en la etapa 2, cc-lfs, para compilar dichas bibliotecas.

• En general, cc-lfs no puede ejecutarse en pc (el sistema host). Aunque los tripletes para pc y lfs son compatibles entre sí, un ejecutable para lfs debe depender de glibc-2.41. La distribución anfitriona puede utilizar una implementación diferente de libc (por ejemplo, musl) o una versión anterior de glibc (por ejemplo, glibc-2.13).

• Aunque cc-lfs pueda ejecutarse en PC, su uso en PC podría generar el riesgo de enlazar con las bibliotecas de PC, ya que cc-lfs es un compilador nativo.

Por lo tanto, al compilar la etapa 2 de gcc, indicamos al sistema de compilación que reconstruya libgcc y libstdc++ con cc1, pero enlazamos libstdc++ con la libgcc recién reconstruida en lugar de con la compilación anterior y degradada. Esto hace que libstdc++ reconstruida sea completamente funcional.

En el Capítulo 8 (o "etapa 3"), se compilan todos los paquetes necesarios para el sistema LFS. Incluso si un paquete ya se ha instalado en el sistema LFS en un capítulo anterior, lo reconstruimos. La razón principal para reconstruir estos paquetes es su estabilidad: si reinstalamos un paquete LFS en un sistema LFS completo, el contenido reinstalado del paquete debe ser el mismo que el contenido del mismo paquete cuando se instaló por primera vez en el Capítulo 8. Los paquetes temporales instalados en los Capítulos 6 o 7 no pueden cumplir este requisito, ya que algunos se compilan sin dependencias opcionales, y autoconf no puede realizar algunas comprobaciones de características en el Capítulo 6 debido a la compilación cruzada, lo que provoca que los paquetes temporales carezcan de características opcionales o utilicen rutinas de código deficientes. Además, una razón menor para reconstruir los paquetes es ejecutar las suites de pruebas.

Otros detalles del procedimiento

El compilador cruzado se instalará en un directorio separado, $LFS/tools, ya que no formará parte del sistema final.

Binutils se instala primero porque las ejecuciones de configuración de gcc y glibc realizan diversas pruebas de funcionalidad en el ensamblador y el enlazador para determinar qué funciones del software habilitar o deshabilitar. Esto es más importante de lo que uno podría pensar inicialmente. Una configuración incorrecta de gcc o glibc puede resultar en una cadena de herramientas ligeramente dañada, cuyo impacto podría no ser evidente hasta casi el final de la compilación de una distribución completa. Un fallo en la suite de pruebas generalmente resaltará este error antes de que se realice demasiado trabajo adicional.

Binutils instala su ensamblador y enlazador en dos ubicaciones: $LFS/tools/bin y $LFS/tools/$LFS_TGT/bin. Las herramientas en una ubicación están enlazadas directamente a la otra. Un aspecto importante del enlazador es el orden de búsqueda de sus bibliotecas. Se puede obtener información detallada de **ld** pasándole el indicador --verbose. Por ejemplo, **$LFS_TGT-ld --verbose | grep SEARCH** mostrará las rutas de búsqueda actuales y su orden. (Tenga en cuenta que este ejemplo solo se puede ejecutar como se muestra con la sesión iniciada como el usuario lfs. Si vuelve a esta página más tarde, reemplace **$LFS_TGT-ld** por **ld**).

El siguiente paquete que se instala es gcc. Un ejemplo de lo que se puede ver durante la ejecución de **configure** es:

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

Esto es importante por las razones mencionadas anteriormente. También demuestra que el script de configuración de gcc no busca en los directorios PATH para encontrar las herramientas que se deben usar. Sin embargo, durante la ejecución de gcc, no se utilizan necesariamente las mismas rutas de búsqueda. Para averiguar qué enlazador estándar usará **gcc**, ejecute: **$LFS_TGT-gcc -print-prog-name=ld**.

(De nuevo, elimine el prefijo **$LFS_TGT-** si vuelve a este tema más adelante).

Se puede obtener información detallada de gcc pasándole la opción de línea de comandos -v al compilar un programa. Por ejemplo, **$LFS_TGT-gcc -v example.c** (o sin **$LFS_TGT-** si vuelve más adelante) mostrará información detallada sobre las etapas de preprocesamiento, compilación y ensamblaje, incluyendo las rutas de búsqueda de **gcc** para los encabezados incluidos y su orden.

A continuación: encabezados de la API de Linux depurados. Estos permiten que la biblioteca estándar de C (glibc) interactúe con las funciones que proporcionará el kernel de Linux.

A continuación viene glibc. Las consideraciones más importantes para compilar glibc son el compilador, las herramientas binarias y los encabezados del kernel. El compilador y las herramientas binarias generalmente no son un problema, ya que glibc siempre usará los relacionados con el parámetro --host pasado a su script de configuración. Por ejemplo, en nuestro caso, el compilador será **$LFS_TGT-gcc** y la herramienta **readelf** será **$LFS_TGT-readelf**. Las cabeceras del kernel pueden ser un poco más complejas. Por lo tanto, no nos arriesgamos y utilizamos la opción de configuración disponible para garantizar la selección correcta. Tras ejecutar la **configuración**, revise el contenido del archivo config.make en el directorio de compilación para obtener todos los detalles importantes. Estos elementos resaltan un aspecto importante del paquete glibc: es muy autosuficiente en cuanto a su maquinaria de compilación y, por lo general, no depende de los valores predeterminados de la cadena de herramientas.

Como se mencionó anteriormente, a continuación se compila la biblioteca estándar de C++, seguida, en el Capítulo 6, por otros programas que deben compilarse de forma cruzada para romper las dependencias circulares en tiempo de compilación. El paso de instalación de todos estos paquetes utiliza la variable DESTDIR para forzar la instalación en el sistema de archivos LFS.

Al final del Capítulo 6 se instala el compilador nativo de LFS. Primero se construye binutils-pass2, en el mismo directorio DESTDIR que los demás programas. Luego, se construye el segundo paso de gcc, omitiendo algunas bibliotecas no críticas. Debido a una lógica extraña en el script de configuración de gcc, CC_FOR_TARGET termina como cc cuando el host es el mismo que el destino, pero diferente del sistema de compilación. Por eso, CC_FOR_TARGET=$LFS_TGT-gcc se declara explícitamente como una de las opciones de configuración.

Al entrar en el entorno chroot en el Capítulo 7, se realizan las instalaciones temporales de los programas necesarios para el correcto funcionamiento de la cadena de herramientas. A partir de este punto, la cadena de herramientas principal es autocontenida y autoalojada.

En el Capítulo 8, se construyen, prueban e instalan las versiones finales de todos los paquetes necesarios para un sistema completamente funcional.

Instrucciones generales de compilación

Precaución

Durante el ciclo de desarrollo de LFS, las instrucciones del libro suelen modificarse para adaptarse a una actualización de paquete o aprovechar las nuevas características de los paquetes actualizados. Mezclar las instrucciones de diferentes versiones del libro de LFS puede causar fallos sutiles. Este tipo de problema suele deberse a la reutilización de algún script creado para una versión anterior de LFS. Se desaconseja encarecidamente dicha reutilización. Si por cualquier motivo reutiliza scripts de una versión anterior de LFS, deberá tener mucho cuidado de actualizarlos para que coincidan con la versión actual del libro de LFS.

A continuación, se indican algunos aspectos que debe saber sobre la compilación de cada paquete:

• Varios paquetes se parchean antes de la compilación, pero solo cuando el parche es necesario para solucionar un problema.

A menudo se necesita un parche tanto en el capítulo actual como en los siguientes, pero a veces, cuando el mismo paquete se compila más de una vez, el parche no es necesario de inmediato. Por lo tanto, no se preocupe si parece que faltan las instrucciones para un parche descargado. También pueden aparecer mensajes de advertencia sobre desplazamiento o fuzz al aplicar un parche. No se preocupe por estas advertencias; el parche se aplicó correctamente.

• Durante la compilación de la mayoría de los paquetes, aparecerán algunas advertencias en la pantalla. Estas son normales y se pueden ignorar sin problemas. Estas advertencias suelen referirse al uso obsoleto, pero no inválido, de la sintaxis de C o C++. Los estándares de C cambian con frecuencia y algunos paquetes aún no se han actualizado. Esto no es un problema grave, pero sí provoca la aparición de las advertencias.

• Compruebe una última vez que la variable de entorno LFS esté configurada correctamente:

```
echo $LFS
```

Asegúrese de que la salida muestre la ruta al punto de montaje de la partición LFS, que es /mnt/lfs, usando nuestro ejemplo.

• Finalmente, es importante destacar dos puntos importantes:

Importante

Las instrucciones de compilación asumen que los requisitos del sistema host, incluidos los enlaces simbólicos, se han configurado correctamente:
• bash es el shell en uso.
• sh es un enlace simbólico a bash. • /usr/bin/awk es un enlace simbólico a gawk.
• /usr/bin/yacc es un enlace simbólico a bison o a un pequeño script que lo ejecuta.

Importante

A continuación, se presenta una sinopsis del proceso de compilación.

1. Coloque todas las fuentes y parches en un directorio accesible desde el entorno chroot, como /mnt/lfs/sources/.

2. Vaya al directorio /mnt/lfs/sources/.

3. Para cada paquete:

   a. Extraiga el paquete que se va a compilar con el programa tar. En los capítulos 5 y 6, asegúrese de ser el usuario lfs al extraer el paquete.
No utilice ningún otro método que no sea el comando tar para extraer el código fuente. En particular, usar el comando cp -R para copiar el árbol de código fuente en otro lugar puede destruir las marcas de tiempo del árbol de código fuente y provocar un fallo en la compilación.

  b. Vaya al directorio creado al extraer el paquete.

  c. Siga las instrucciones para compilar el paquete.

  d. Vuelva al directorio de fuentes una vez completada la compilación.

  e. Elimine el directorio de fuentes extraído a menos que se le indique lo contrario.

Capítulo 5. Compilación de una cadena de herramientas cruzadas

5.1. Introducción

Este capítulo muestra cómo compilar un compilador cruzado y sus herramientas asociadas. Aunque aquí se simula una compilación cruzada, los principios son los mismos que para una cadena de herramientas cruzadas real.

Los programas compilados en este capítulo se instalarán en el directorio $LFS/tools para mantenerlos separados de los archivos que se instalarán en los capítulos siguientes. Las bibliotecas, por otro lado, se instalan en su ubicación final, ya que pertenecen al sistema que queremos compilar.

5.2. Binutils-2.44 - Paso 1

El paquete Binutils contiene un enlazador, un ensamblador y otras herramientas para gestionar archivos objeto.

Tiempo de compilación aproximado: 1 SBU
Espacio en disco requerido: 677 MB

5.2.1. Instalación de Cross Binutils

Nota

Revise las notas en la sección titulada Instrucciones generales de compilación. Comprender las notas marcadas como importantes puede ahorrarle muchos problemas más adelante.

Es importante que Binutils sea el primer paquete compilado, ya que tanto Glibc como GCC realizan varias pruebas en el enlazador y ensamblador disponibles para determinar cuáles de sus propias funciones habilitar.

La documentación de Binutils recomienda compilarlo en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Nota

Para que los valores de SBU que se indican en el resto del libro sean útiles, mida el tiempo que tarda en compilarse este paquete desde la configuración hasta la primera instalación, incluyendo esta. Para lograr esto fácilmente, encierre los comandos en un comando de tiempo como este: time { ../configure ... && make && make install; }.

Ahora prepare Binutils para la compilación:

```
../configure --prefix=$LFS/tools \
             --with-sysroot=$LFS \
             --target=$LFS_TGT   \
             --disable-nls       \
             --enable-gprofng=no \
             --disable-werror    \
             --enable-new-dtags  \
             --enable-default-hash-style=gnu
```

Significado de las opciones de configuración:

`--prefix=$LFS/tools`

Esto indica al script de configuración que se prepare para instalar los programas de Binutils en el directorio $LFS/tools.

--with-sysroot=$LFS

Para la compilación cruzada, esto indica al sistema de compilación que busque en $LFS las bibliotecas del sistema de destino según sea necesario. --target=$LFS_TGT

Dado que la descripción de la máquina en la variable LFS_TGT es ligeramente diferente del valor devuelto por el script config.guess, esta opción indicará al script de configuración que ajuste el sistema de compilación de binutil para la creación de un enlazador cruzado.

--disable-nls

Esto deshabilita la internacionalización, ya que i18n no es necesario para las herramientas temporales.

--enable-gprofng=no

Esto deshabilita la creación de gprofng, que no es necesario para las herramientas temporales.

--disable-werror

Esto evita que la compilación se detenga si hay advertencias del compilador del host.

--enable-new-dtags

Esto hace que el enlazador use la etiqueta "runpath" para incrustar rutas de búsqueda de bibliotecas en ejecutables y bibliotecas compartidas, en lugar de la etiqueta tradicional "rpath". Esto facilita la depuración de ejecutables enlazados dinámicamente y soluciona posibles problemas en el conjunto de pruebas de algunos paquetes.

--enable-default-hash-style=gnu

Por defecto, el enlazador generaría tanto la tabla hash de estilo GNU como la tabla hash ELF clásica para bibliotecas compartidas y ejecutables enlazados dinámicamente. Las tablas hash solo están diseñadas para que un enlazador dinámico realice la búsqueda de símbolos. En LFS, el enlazador dinámico (proporcionado por el paquete Glibc) siempre usará la tabla hash de estilo GNU, que es más rápida de consultar. Por lo tanto, la tabla hash ELF clásica es completamente inútil. Esto hace que el enlazador solo genere la tabla hash estilo GNU por defecto, lo que evita perder tiempo generando la tabla hash ELF clásica al compilar los paquetes, o desperdiciando espacio en disco para almacenarla.

Continúe compilando el paquete:

```
make
```

Instalar el paquete:

```
make install
```

Los detalles de este paquete se encuentran en la Sección 8.20.2, “Contenido de Binutils”.

5.3. GCC-14.2.0 - Paso 1

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

Tiempo de compilación aproximado: 3,2 SBU
Espacio en disco requerido: 4,8 GB

5.3.1. Instalación de Cross GCC

GCC requiere los paquetes GMP, MPFR y MPC. Dado que estos paquetes podrían no estar incluidos en su distribución, se compilarán con GCC. Descomprima cada paquete en el directorio de origen de GCC y renombre los directorios resultantes para que los procedimientos de compilación de GCC los utilicen automáticamente:

Nota

Este capítulo suele dar lugar a malentendidos. Los procedimientos son los mismos que en los demás capítulos, como se explicó anteriormente (Instrucciones de compilación de paquetes). Primero, extraiga el archivo tar gcc-14.2.0 del directorio de origen y luego acceda al directorio creado. Solo entonces debe continuar con las instrucciones a continuación.

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

En hosts x86_64, configure el nombre de directorio predeterminado para las bibliotecas de 64 bits como “lib”:

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64

  ;;
esac
```

La documentación de GCC recomienda compilar GCC en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Preparar GCC para la compilación:

```
../configure                  \
    --target=$LFS_TGT         \
    --prefix=$LFS/tools       \
    --with-glibc-version=2.41 \
    --with-sysroot=$LFS       \
    --with-newlib             \
    --without-headers         \
    --enable-default-pie      \
    --enable-default-ssp      \
    --disable-nls             \
    --disable-shared          \
    --disable-multilib        \
    --disable-threads         \
    --disable-libatomic       \
    --disable-libgomp         \
    --disable-libquadmath     \
    --disable-libssp          \
    --disable-libvtv          \
    --disable-libstdcxx       \
    --enable-languages=c,c++
```

Significado de las opciones de configuración:

`--with-glibc-version=2.41`

Esta opción especifica la versión de Glibc que se usará en el Destino. No es relevante para la biblioteca libc de la distribución anfitriona, ya que todo lo compilado por pass1 de GCC se ejecutará en el entorno chroot, que está aislado de la biblioteca libc de la distribución anfitriona.

`--with-newlib`

Dado que aún no hay una biblioteca C funcional disponible, esto garantiza que la constante inhibitor_libc se defina al compilar libgcc. Esto evita la compilación de código que requiera compatibilidad con libc.

`--without-headers`

Al crear un compilador cruzado completo, GCC requiere encabezados estándar compatibles con el sistema de destino. Para nuestros propósitos, estos encabezados no serán necesarios. Esta opción impide que GCC los busque.

`--enable-default-pie y --enable-default-ssp`

Estas opciones permiten a GCC compilar programas con algunas características de seguridad reforzadas (más información sobre ellas en la nota sobre PIE y SSP en el capítulo 8) de forma predeterminada. No son estrictamente necesarias en esta etapa, ya que el compilador solo producirá ejecutables temporales. Pero es más limpio que los paquetes temporales sean lo más parecidos posible a los finales.

--disable-shared

Esta opción obliga a GCC a enlazar sus bibliotecas internas estáticamente. Esto es necesario porque las bibliotecas compartidas requieren Glibc, que aún no está instalada en el sistema de destino.

--disable-multilib

En x86_64, LFS no admite una configuración multilib. Esta opción es inofensiva para x86.

--disable-threads, --disable-libatomic, --disable-libgomp, --disable-libquadmath, --disable-libssp, --disable-libvtv, --disable-libstdcxx

Estas opciones deshabilitan la compatibilidad con subprocesos, libatomic, libgomp, libquadmath, libssp, libvtv y la biblioteca estándar de C++, respectivamente. Estas funciones pueden fallar al compilar un compilador cruzado y no son necesarias para la tarea de compilación cruzada de la biblioteca temporal libc.

--enable-languages=c,c++

Esta opción garantiza que solo se compilen los compiladores de C y C++. Estos son los únicos lenguajes necesarios ahora.

Compile GCC ejecutando:

```
make
```

Instale el paquete:

```
make install
```

Esta compilación de GCC ha instalado un par de encabezados de sistema internos. Normalmente, uno de ellos, limits.h, incluiría a su vez el encabezado limits.h del sistema correspondiente; en este caso, $LFS/usr/include/limits.h. Sin embargo, en el momento de esta compilación de GCC, $LFS/usr/include/limits.h no existía, por lo que el encabezado interno que se acaba de instalar es un archivo parcial e independiente que no incluye las funciones extendidas del encabezado del sistema. Esto es suficiente para compilar Glibc, pero el encabezado interno completo se necesitará más adelante. Cree una versión completa del encabezado interno con un comando idéntico a lo que el sistema de compilación GCC realiza en circunstancias normales:

Nota

El siguiente comando muestra un ejemplo de sustitución de comandos anidada con dos métodos: comillas invertidas y una construcción $(). Podría reescribirse con el mismo método para ambas sustituciones, pero se muestra así para demostrar cómo combinarlas. Generalmente, se prefiere el método $().

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

Los detalles de este paquete se encuentran en la Sección 8.29.2, “Contenido de GCC”.

5.4. Cabeceras de la API de Linux-6.13.4

Las cabeceras de la API de Linux (en linux-6.13.4.tar.xz) exponen la API del kernel para que Glibc los use.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 1,6 GB

5.4.1. Instalación de las cabeceras de la API de Linux

El kernel de Linux necesita exponer una interfaz de programación de aplicaciones (API) para que la biblioteca C del sistema (Glibc en LFS) la utilice. Esto se realiza mediante la limpieza de varios archivos de cabecera en lenguaje C incluidos en el archivo tarball del kernel de Linux.

Asegúrese de que no haya archivos obsoletos incrustados en el paquete:

```
make mrproper
```

Ahora extraiga los encabezados del kernel visibles para el usuario del código fuente. El destino de creación recomendado "headers_install" no se puede usar porque requiere rsync, que podría no estar disponible. Los encabezados se colocan primero en ./usr y luego se copian en la ubicación necesaria. crear encabezados

```
make headers
find usr/include -type f ! -name '*.h' -delete
cp -rv usr/include $LFS/usr
```

5.4.2. Contenido de las cabeceras de la API de Linux

Cabeceras instaladas: /usr/include/asm/*.h, /usr/include/asm-generic/*.h, /usr/include/drm/*.h, /usr/include/linux/*.h, /usr/include/misc/*.h, /usr/include/mtd/*.h, /usr/include/rdma/*.h, /usr/include/scsi/*.h, /usr/include/sound/*.h, /usr/include/video/*.h y /usr/include/xen/*.h.

Directorios instalados: /usr/include/asm, /usr/include/asm-generic, /usr/include/drm, /usr/include/linux, /usr/include/misc /usr/include/mtd, /usr/include/rdma, /usr/include/scsi, /usr/include/sound, /usr/include/video y /usr/include/xen.

Descripciones breves

/usr/include/asm/*.h	Cabeceras ASM de la API de Linux.
/usr/include/asm-generic/*.h	Cabeceras genéricas ASM de la API de Linux.
/usr/include/drm/*.h	Cabeceras DRM de la API de Linux.
/usr/include/linux/*.h	Cabeceras Linux de la API de Linux.
/usr/include/misc/*.h	Cabeceras misceláneas de la API de Linux.

/usr/include/mtd/*.h	Cabeceras MTD de la API de Linux.
/usr/include/rdma/*.h	Cabeceras RDMA de la API de Linux.
/usr/include/scsi/*.h	Cabeceras SCSI de la API de Linux Encabezados.
/usr/include/sound/*.h	Cabeceras de sonido de la API de Linux.
/usr/include/video/*.h	Cabeceras de vídeo de la API de Linux.
/usr/include/xen/*.h	Cabeceras de Xen de la API de Linux.

5.5. Glibc-2.41

El paquete Glibc contiene la biblioteca principal de C. Esta biblioteca proporciona las rutinas básicas para asignar memoria, buscar directorios, abrir y cerrar archivos, leer y escribir archivos, manejar cadenas, buscar patrones, realizar operaciones aritméticas, etc.

Tiempo de compilación aproximado: 1.4 SBU
Espacio en disco requerido: 850 MB

5.5.1. Instalación de Glibc

Primero, cree un enlace simbólico para la compatibilidad con LSB. Además, para x86_64, cree un enlace simbólico de compatibilidad necesario para el correcto funcionamiento del cargador dinámico de bibliotecas:

```
case $(uname -m) in
    i?86)   ln -sfv ld-linux.so.2 $LFS/lib/ld-lsb.so.3
    ;;
    x86_64) ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64
            ln -sfv ../lib/ld-linux-x86-64.so.2 $LFS/lib64/ld-lsb-x86-64.so.3
    ;;
esac
```

Nota

El comando anterior es correcto. El comando ln tiene varias versiones sintácticas, así que asegúrese de consultar la información de coreutils ln y ln(1) antes de informar de lo que podría parecer un error.

Algunos de los programas Glibc utilizan el directorio /var/db no compatible con FHS para almacenar sus datos de tiempo de ejecución. Aplique el siguiente parche para que dichos programas almacenen sus datos de ejecución en ubicaciones compatibles con FHS:

```
patch -Np1 -i ../glibc-2.41-fhs-1.patch
```

La documentación de Glibc recomienda compilar Glibc en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Asegúrese de que las utilidades ldconfig y sln estén instaladas en /usr/sbin:

```
echo "rootsbindir=/usr/sbin" > configparms
```

A continuación, prepare Glibc para la compilación:

```
../configure \
      --prefix=/usr                     \
      --host=$LFS_TGT                   \
      --build=$(../scripts/config.guess) \
      --enable-kernel=5.4               \
      --with-headers=$LFS/usr/include   \
      --disable-nscd                    \
      libc_cv_slibdir=/usr/lib
```

Significado de las Opciones de configuración:

--host=$LFS_TGT, --build=$(../scripts/config.guess)

El efecto combinado de estas opciones es que el sistema de compilación de Glibc se configura para la compilación cruzada, utilizando el enlazador cruzado y el compilador cruzado en $LFS/tools.

--enable-kernel=5.4

Esto indica a Glibc que compile la biblioteca con compatibilidad con kernels Linux 5.4 y posteriores. Las soluciones alternativas para kernels anteriores no están habilitadas.

--with-headers=$LFS/usr/include

Esto indica a Glibc que se compile automáticamente con las cabeceras instaladas recientemente en el directorio $LFS/usr/include, para que sepa exactamente qué características tiene el kernel y pueda optimizarse en consecuencia.

libc_cv_slibdir=/usr/lib

Esto garantiza que la biblioteca se instale en /usr/lib en lugar del directorio predeterminado /lib64 en equipos de 64 bits.

--disable-nscd

No cree el demonio de caché del servicio de nombres que ya no se utiliza.

Durante esta etapa, podría aparecer la siguiente advertencia:

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

El programa msgfmt faltante o incompatible generalmente es inofensivo. Este programa msgfmt forma parte del paquete Gettext, que la distribución del host debería proporcionar.

Nota

Se han reportado errores al compilar este paquete como "make paralelo". Si esto ocurre, vuelva a ejecutar el comando make con la opción -j1.

Compilar el paquete:

```
make
```

Instalar el paquete:

Advertencia

Si LFS no está configurado correctamente y, a pesar de las recomendaciones, compila como root, el siguiente comando instalará la Glibc recién compilada en su

sistema host, lo que casi con seguridad la dejará inutilizable. Por lo tanto, verifique que el entorno esté configurado correctamente y que no sea root antes de ejecutar el siguiente comando.

```
make DESTDIR=$LFS install
```

Significado de la opción make install:

DESTDIR=$LFS

Casi todos los paquetes utilizan la variable de make DESTDIR para definir la ubicación donde se debe instalar el paquete. Si no está configurada, se establece por defecto en el directorio raíz (/). Aquí especificamos que el paquete se instala en $LFS, que se convertirá en el directorio raíz en la Sección 7.4, “Ingreso al entorno Chroot”.

Corrija una ruta fija al cargador de ejecutables en el script **ldd**:

```
sed '/RTLDLIST=/s@/usr@@g' -i $LFS/usr/bin/ldd
```

Precaución

En este punto, es fundamental detener el proceso y asegurarse de que las funciones básicas (compilación y enlazado) de la nueva cadena de herramientas funcionen correctamente. Para realizar una comprobación de seguridad, ejecute los siguientes comandos:

```
echo 'int main(){}' | $LFS_TGT-gcc -xc -
readelf -l a.out | grep ld-linux
```

Si todo funciona correctamente, no debería haber errores y la salida del último comando tendrá el siguiente formato:

```
[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Tenga en cuenta que para máquinas de 32 bits, el nombre del intérprete será /lib/ld-linux.so.2.
Si la salida no es la mostrada arriba, o no hay salida alguna, algo va mal. Investigue y vuelva a seguir los pasos para encontrar el problema y corregirlo. Este problema debe resolverse antes de continuar.
Una vez que todo esté correcto, limpie el archivo de prueba:

```
rm -v a.out
```

Nota

La compilación de los paquetes en el siguiente capítulo servirá como comprobación adicional de que la cadena de herramientas se ha compilado correctamente. Si algún paquete, especialmente Binutils-pass2 o GCC-pass2, no se compila, es señal de que algo ha fallado con las instalaciones anteriores de Binutils, GCC o Glibc.

Los detalles de este paquete se encuentran en la Sección 8.5.3, “Contenido de Glibc”.

5.6. Libstdc++ desde GCC-14.2.0

Libstdc++ es la biblioteca estándar de C++. Es necesaria para compilar código C++ (parte de GCC está escrita en C++), pero tuvimos que posponer su instalación al compilar gcc-pass1 porque Libstdc++ depende de Glibc, que aún no estaba disponible en el directorio de destino.

Tiempo de compilación aproximado: 0,2 SBU
Espacio en disco requerido: 850 MB

5.6.1. Instalación de Libstdc++ de destino

Nota

Libstdc++ forma parte del código fuente de GCC. Primero debe descomprimir el archivo tar de GCC y acceder al directorio gcc-14.2.0.

Cree un directorio de compilación independiente para Libstdc++ e introdúzcalo:

```
mkdir -v build
cd       build
```

Prepare Libstdc++ para la compilación:

```
../libstdc++-v3/configure          \
    --host=$LFS_TGT                \
    --build=$(../config.guess)     \
    --prefix=/usr                  \
    --disable-multilib             \
    --disable-nls                  \
    --disable-libstdcxx-pch        \
    --with-gxx-include-dir=/tools/$LFS_TGT/include/c++/14.2.0
```

Significado de las opciones de configuración:

`--host=...`

Especifica que se debe usar el compilador cruzado que acabamos de compilar en lugar del de /usr/bin.

`--disable-libstdcxx-pch`

Esta opción impide la instalación de archivos de inclusión precompilados, que no son necesarios en esta etapa.

`--with-gxx-include-dir=/tools/$LFS_TGT/include/c++/14.2.0`

Esto especifica el directorio de instalación de los archivos de inclusión. Dado que Libstdc++ es la biblioteca estándar de C++ para LFS, este directorio debe coincidir con la ubicación donde el compilador de C++ (**$LFS_TGT-g++**) buscaría los archivos de inclusión estándar de C++. En una compilación normal, esta información se pasa automáticamente a las opciones de **configuración** de Libstdc++ desde el directorio de nivel superior. En nuestro caso, esta información debe proporcionarse explícitamente. El compilador de C++ antepondrá la ruta raíz

del sistema $LFS (especificada al compilar GCC-pass1) a la ruta de búsqueda del archivo de inclusión, por lo que buscará en $LFS/tools/$LFS_TGT/include/c++/14.2.0. La combinación de la variable *DESTDIR* (en el comando `**make install**` a continuación) y esta opción hace que las cabeceras se instalen allí.

Compile Libstdc++ ejecutando:

```
make
```

Instale la biblioteca:

```
make DESTDIR=$LFS install
```

Elimina los archivos de libtool, ya que son perjudiciales para la compilación cruzada:

```
rm -v $LFS/usr/lib/lib{stdc++{,exp,fs},supc++}.la
```

Los detalles de este paquete se encuentran en la Sección 8.29.2, “Contenido de GCC”.

Capítulo 6. Herramientas temporales para compilación cruzada

6.1. Introducción

Este capítulo muestra cómo compilar de forma cruzada utilidades básicas utilizando la cadena de herramientas cruzadas recién compilada. Estas utilidades se instalan en su ubicación final, pero aún no se pueden utilizar. Las tareas básicas aún dependen de las herramientas del host. Sin embargo, las bibliotecas instaladas se utilizan al enlazar.

El uso de las utilidades será posible en el siguiente capítulo, tras acceder al entorno "chroot". Sin embargo, todos los paquetes compilados en este capítulo deben compilarse antes de realizarlo. Por lo tanto, aún no podemos ser independientes del sistema *host*. Una vez más, recordemos que una configuración incorrecta de *LFS*, junto con la compilación como *root*, puede inutilizar el equipo.

Todo este capítulo debe completarse con el usuario *lfs*, con el entorno descrito en la Sección 4.4, "Configuración del entorno".

6.2. M4-1.4.19

El paquete M4 contiene un procesador de macros.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 32 MB

6.2.1. Instalación de M4

Preparar M4 para la compilación:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.13.2, “Contenido de M4”.

6.3. Ncurses-6.5

El paquete Ncurses contiene bibliotecas para la gestión de pantallas de caracteres independiente de la terminal.

Tiempo de compilación aproximado: 0,4 SBU
Espacio en disco requerido: 53 MB

6.3.1. Instalación de Ncurses

Primero, ejecute los siguientes comandos para compilar el programa "tic" en el host de compilación:

```
mkdir build
pushd build
  ../configure AWK=gawk
  make -C include
  make -C progs tic
popd
```

Preparar Ncurses para la compilación:

```
./configure --prefix=/usr                \
            --host=$LFS_TGT              \
            --build=$(./config.guess)    \
            --mandir=/usr/share/man      \
            --with-manpage-format=normal \
            --with-shared                \
            --without-normal             \
            --with-cxx-shared            \
            --without-debug              \
            --without-ada                \
            --disable-stripping          \
            AWK=gawk
```

Significado de las nuevas opciones de configuración:

`--with-manpage-format=normal`

Esto evita que Ncurses instale páginas de manual comprimidas, lo que puede ocurrir si la distribución del host ya las tiene.

`--with-shared`

Esto hace que Ncurses compile e instale bibliotecas de C compartidas.

`--without-normal`

Esto impide que Ncurses compile e instale bibliotecas de C estáticas.

`--without-debug`

Esto impide que Ncurses compile e instale bibliotecas de depuración.

--with-cxx-shared

Esto hace que Ncurses compile e instale enlaces de C++ compartidos. También impide que compile e instale enlaces de C++ estáticos.

--without-ada

Esto garantiza que Ncurses no compile compatibilidad con el compilador de Ada, que puede estar presente en el host, pero no estará disponible una vez que entremos en el entorno **chroot**.

--disable-stripping

Este modificador evita que el sistema de construcción utilice el programa **strip** del sistema host. Usar herramientas del host en programas compilados de forma cruzada puede provocar fallos.

AWK=gawk

Esta opción impide que el sistema de compilación use el programa **mawk** del host. Algunas versiones de **mawk** pueden provocar que este paquete no se compile.

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS TIC_PATH=$(pwd)/build/progs/tic install
ln -sv libncursesw.so $LFS/usr/lib/libncurses.so
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i $LFS/usr/include/curses.h
```

Significado de las opciones de instalación:

TIC_PATH=$(pwd)/build/progs/tic

Necesitamos pasar la ruta del programa tic recién compilado que se ejecuta en el equipo de compilación para que la base de datos de la terminal pueda crearse sin errores.

ln -sv libncursesw.so $LFS/usr/lib/libncurses.so

La biblioteca libncurses.so es necesaria para algunos paquetes que crearemos próximamente. Creamos este enlace simbólico para usar libncursesw.so como reemplazo.

sed -e 's/^#if.*XOPEN.*$/#if 1/' ...

El archivo de encabezado curses.h contiene la definición de varias estructuras de datos de Ncurses. Con diferentes definiciones de macros de preprocesador, se pueden usar dos conjuntos diferentes de definiciones de estructuras de datos: la definición de 8 bits es compatible con libncurses.so y la definición de caracteres anchos es compatible con libncursesw.so. Dado que usamos libncursesw.so como reemplazo de libncurses.so, edite el archivo de encabezado para que siempre use la definición de estructura de datos de caracteres anchos compatible con libncursesw.so.

Los detalles de este paquete se encuentran en la Sección 8.30.2, “Contenido de Ncurses”.

6.4. Bash-5.2.37

El paquete Bash contiene Bourne-Again Shell.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 68 MB

6.4.1. Instalación de Bash

Preparar Bash para la compilación:

```
./configure --prefix=/usr                      \
            --build=$(sh support/config.guess) \
            --host=$LFS_TGT                    \
            --without-bash-malloc
```

Significado de las opciones de configuración:

`--without-bash-malloc`

Esta opción desactiva el uso de la función de asignación de memoria de Bash (malloc), que se sabe que causa fallos de segmentación. Al desactivar esta opción, Bash utilizará las funciones malloc de Glibc, que son más estables. Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Crear un enlace para los programas que usan sh como shell:

```
ln -sv bash $LFS/bin/sh
```

Los detalles de este paquete se encuentran en la Sección 8.36.2, “Contenido de Bash”.

6.5. Coreutils-9.6

El paquete Coreutils contiene las utilidades básicas que necesita cualquier sistema operativo.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 181 MB

6.5.1. Instalación de Coreutils

Preparar Coreutils para la compilación:

```
./configure --prefix=/usr \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess) \
            --enable-install-program=hostname \
            --enable-no-install-program=kill,uptime
```

Significado de las opciones de configuración:

`--enable-install-program=hostname`

Esto permite compilar e instalar el binario **hostname**; está deshabilitado por defecto, pero es requerido por el conjunto de pruebas de Perl.

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Mueva los programas a sus ubicaciones finales previstas. Aunque esto no es necesario en este entorno temporal, debemos hacerlo porque algunos programas codifican las ubicaciones de los ejecutables:

```
mv -v $LFS/usr/bin/chroot              $LFS/usr/sbin
mkdir -pv $LFS/usr/share/man/man8
mv -v $LFS/usr/share/man/man1/chroot.1 $LFS/usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/'                    $LFS/usr/share/man/man8/chroot.8
```

Los detalles de este paquete se encuentran en la Sección 8.58.2, “Contenido de Coreutils”.

6.6. Diffutils-3.11

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

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 35 MB

6.6.1. Instalación de Diffutils

Preparar Diffutils para la compilación:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(./build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.60.2, “Contenido de Diffutils”.

6.7. File-5.46

El paquete File contiene una utilidad para determinar el tipo de uno o más archivos.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 42 MB

6.7.1. Instalación de File

El comando file en el host de compilación debe tener la misma versión que la que estamos compilando para crear el archivo de firma. Ejecute los siguientes comandos para crear una copia temporal del archivo:

```
mkdir build
pushd build
  ../configure --disable-bzlib      \
               --disable-libseccomp \
               --disable-xzlib      \
               --disable-zlib
  make
popd
```

Significado de la nueva opción de configuración:

`--disable-*`

El script de configuración intenta usar algunos paquetes de la distribución del host si existen los archivos de biblioteca correspondientes. Puede causar un error de compilación si existe un archivo de biblioteca, pero no los archivos de encabezado correspondientes. Estas opciones impiden el uso de estas funciones innecesarias del host.

Preparar el archivo para la compilación:

```
./configure --prefix=/usr --host=$LFS_TGT –build=$(./config.guess)
```

Compilar el paquete:

```
make FILE_COMPILE=$(pwd)/build/src/file
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Eliminar el archivo libtool, ya que puede ser perjudicial para la compilación cruzada:

```
rm -v $LFS/usr/lib/libmagic.la
```

Los detalles de este paquete se encuentran en la Sección 8.11.2, “Contenido del archivo”.

6.8. Findutils-4.10.0

El paquete Findutils contiene programas para buscar archivos. Se proporcionan programas para buscar en todos los archivos de un árbol de directorios y para crear, mantener y buscar en una base de datos (a menudo más rápido que la búsqueda recursiva, pero poco fiable a menos que la base de datos se haya actualizado recientemente). Findutils también incluye el programa **xargs**, que permite ejecutar un comando específico en cada archivo seleccionado en una búsqueda.

Tiempo de compilación aproximado: 0,2 SBU
Espacio en disco requerido: 48 MB

6.8.1. Instalación de Findutils

Preparar Findutils para la compilación:

```
./configure --prefix=/usr                     \
            --localstatedir=/var/lib/locate   \
            --host=$LFS_TGT                   \
            --build=$(build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.62.2, “Contenido de Findutils”.

6.9. Gawk-5.3.1

El paquete Gawk contiene programas para manipular archivos de texto.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 47 MB

6.9.1. Instalación de Gawk

Primero, asegúrese de que no se instalen archivos innecesarios:

```
sed -i 's/extras//' Makefile.in
```

Prepare Gawk para la compilación:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.61.2, “Contenido de Gawk”.

6.10. Grep-3.11

El paquete Grep contiene programas para buscar en el contenido de los archivos.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 27 MB

6.10.1. Instalación de Grep

Preparar Grep para la compilación:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(./build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.35.2, “Contenido de Grep”.

6.11. Gzip-1.13

El paquete Gzip contiene programas para comprimir y descomprimir archivos.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 11 MB

6.11.1. Instalación de Gzip

Preparar Gzip para la compilación:

```
./configure --prefix=/usr --host=$LFS_TGT
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.65.2, “Contenido de Gzip”.

6.12. Make-4.4.1

El paquete Make contiene un programa para controlar la generación de ejecutables y otros archivos no fuente de un paquete a partir de los archivos fuente.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 15 MB

6.12.1. Instalación de Make

Preparar Make para la compilación:

```
./configure --prefix=/usr   \
            --without-guile \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Significado de la nueva opción de configuración:

`--without-guile`

Aunque se realiza una compilación cruzada, configure intenta usar guile del host de compilación si lo encuentra. Esto provoca un error en la compilación, por lo que esta opción impide su uso. Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.69.2, “Contenido de Make”.

6.13. Parche-2.7.6

El paquete Parche contiene un programa para modificar o crear archivos mediante la aplicación de un archivo de parche, generalmente creado por el programa **diff**.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 12 MB

6.13.1. Instalación del parche

Preparar el parche para la compilación:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.70.2, “Contenido del parche”.

6.14. Sed-4.9

El paquete Sed contiene un editor de flujos.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 21 MB

6.14.1. Instalación de Sed

Preparar Sed para la compilación:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(./build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.31.2, “Contenido de Sed”.

6.15. Tar-1.35

El paquete Tar permite crear archivos tar, así como realizar otras manipulaciones de archivos. Tar se puede usar en archivos creados previamente para extraer archivos, almacenar archivos adicionales o actualizar o listar archivos ya almacenados.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 42 MB

6.15.1. Instalación de Tar

Preparar Tar para la compilación:

```
./configure --prefix=/usr   \
            --host=$LFS_TGT \
            --build=$(build-aux/config.guess)
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Los detalles de este paquete se encuentran en la Sección 8.71.2, “Contenido de Tar”.

6.16. Xz-5.6.4

El paquete Xz contiene programas para comprimir y descomprimir archivos. Ofrece funciones para los formatos de compresión lzma y xz, que son más recientes. Comprimir archivos de texto con xz ofrece un mejor porcentaje de compresión que con los comandos tradicionales gzip o bzip2.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en disco requerido: 21 MB

6.16.1. Instalación de Xz

Preparar Xz para la compilación:

```
./configure --prefix=/usr                     \
            --host=$LFS_TGT                   \
            --build=$(build-aux/config.guess) \
            --disable-static                  \
            --docdir=/usr/share/doc/xz-5.6.4
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Eliminar el archivo libtool, ya que es perjudicial para la compilación cruzada:

```
rm -v $LFS/usr/lib/liblzma.la
```

Los detalles de este paquete se encuentran en la Sección 8.8.2, “Contenido de Xz”.

6.17. Binutils-2.44 - Paso 2

El paquete Binutils contiene un enlazador, un ensamblador y otras herramientas para gestionar archivos objeto.

Tiempo de compilación aproximado: 0,4 SBU
Espacio en disco requerido: 539 MB

6.17.1. Instalación de Binutils

El sistema de compilación de Binutils se basa en una copia de libtool incluida para enlazar con bibliotecas estáticas internas, pero las copias de libiberty y zlib incluidas en el paquete no usan libtool. Esta inconsistencia puede provocar que los binarios generados se enlacen por error con bibliotecas de la distribución anfitriona. Solución alternativa:

```
sed '6031s/$add_dir//' -i ltmain.sh
```

Crear de nuevo un directorio de compilación independiente:

```
mkdir -v build
cd       build
```

Preparar Binutils para la compilación:

```
../configure                   \
    --prefix=/usr              \
    --build=$(../config.guess) \
    --host=$LFS_TGT            \
    --disable-nls              \
    --enable-shared            \
    --enable-gprofng=no        \
    --disable-werror           \
    --enable-64-bit-bfd        \
    --enable-new-dtags         \
    --enable-default-hash-style=gnu
```

Significado de las nuevas opciones de configuración:

`--enable-shared`

Compila libbfd como una biblioteca compartida.

`--enable-64-bit-bfd`

Habilita la compatibilidad con 64 bits (en hosts con tamaños de palabra más pequeños). Puede que esto no sea necesario en sistemas de 64 bits, pero no causa ningún daño.

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make DESTDIR=$LFS install
```

Eliminar los archivos de libtool, ya que son perjudiciales para la compilación cruzada, y eliminar las bibliotecas estáticas innecesarias:

```
rm -v $LFS/usr/lib/lib{bfd,ctf,ctf-nobfd,opcodes,sframe}.{a,la}
```

Los detalles de este paquete se encuentran en la Sección 8.20.2, “Contenido de Binutils”.

6.18. GCC-14.2.0 - Paso 2

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

Tiempo de compilación aproximado: 4,1 SBU
Espacio en disco requerido: 5,5 GB

6.18.1. Instalación de GCC

Al igual que en la primera compilación de GCC, se requieren los paquetes GMP, MPFR y MPC. Descomprima los archivos tar y muévalos a los directorios necesarios:

```
tar -xf ../mpfr-4.2.1.tar.xz
mv -v mpfr-4.2.1 mpfr
tar -xf ../gmp-6.3.0.tar.xz
mv -v gmp-6.3.0 gmp
tar -xf ../mpc-1.3.1.tar.gz
mv -v mpc-1.3.1 mpc
```

Si se compila en x86_64, cambie el nombre del directorio predeterminado para las bibliotecas de 64 bits a "lib":

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
;;
esac
```

Anule la regla de compilación de las cabeceras de libgcc y libstdc++ para permitir la compilación de estas bibliotecas con compatibilidad con subprocesos POSIX:

```
sed '/thread_header =/s/@.*@/gthr-posix.h/' \
    -i libgcc/Makefile.in libstdc++-v3/include/Makefile.in
```

Cree de nuevo un directorio de compilación independiente:

```
mkdir -v build
cd       build
```

Antes de comenzar a compilar GCC, recuerde deshabilitar cualquier variable de entorno que anule los indicadores de optimización predeterminados.

Ahora prepare GCC para la compilación:

```
../configure
    --build=$(../config.guess)                \
    --host=$LFS_TGT                           \
    --target=$LFS_TGT                         \
    LDFLAGS_FOR_TARGET=-L$PWD/$LFS_TGT/libgcc  \
    --prefix=/usr                             \
```

```
    --with-build-sysroot=$LFS                  \
    --enable-default-pie                       \
    --enable-default-ssp                       \
    --disable-nls                              \
    --disable-multilib                         \
    --disable-libatomic                        \
    --disable-libgomp                          \
    --disable-libquadmath                      \
    --disable-libsanitizer                     \
    --disable-libssp                           \
    --disable-libvtv                           \
    --enable-languages=c,c++
```

Significado de las nuevas opciones de configuración:

`--with-build-sysroot=$LFS`

Normalmente, usar --host garantiza que se use un compilador cruzado para compilar GCC, y que ese compilador sepa que debe buscar las cabeceras y bibliotecas en $LFS. Sin embargo, el sistema de compilación de GCC utiliza otras herramientas que desconocen esta ubicación. Esta opción es necesaria para que dichas herramientas encuentren los archivos necesarios en $LFS y no en el host.

`--target=$LFS_TGT`

Estamos compilando GCC de forma cruzada, por lo que es imposible compilar las bibliotecas de destino (libgcc y libstdc++) con los binarios de GCC compilados en este paso; estos binarios no se ejecutarán en el host. El sistema de compilación de GCC intentará usar los compiladores de C y C++ del host como solución alternativa de forma predeterminada. No se admite la compilación de las bibliotecas de destino de GCC con una versión diferente de GCC, por lo que usar los compiladores del host puede provocar un error en la compilación. Este parámetro garantiza que las bibliotecas se compilen con la versión 1 de GCC.

`LDFLAGS_FOR_TARGET=...`

Permite que libstdc++ use la versión libgcc que se compila en esta versión, en lugar de la versión anterior compilada en gcc-pass1. La versión anterior no admite correctamente la gestión de excepciones de C++ porque se compiló sin compatibilidad con libc.

`--disable-libsanitizer`

Desactiva las bibliotecas de ejecución de GCC Sanitizer. No son necesarias para la instalación temporal. En gcc-pass1, esto estaba implícito con --disable-libstdcxx, y ahora podemos pasarlo explícitamente.

Compila el paquete:

```
make
```

Instala el paquete:

```
make DESTDIR=$LFS install
```

Como toque final, crea un enlace simbólico de utilidad. Muchos programas y scripts ejecutan cc en lugar de gcc, lo cual se utiliza para mantener los programas genéricos y, por lo tanto, utilizables en todo tipo de sistemas UNIX donde el compilador GNU C no siempre está instalado. Ejecutar cc permite al administrador del sistema decidir libremente qué compilador de C instalar:

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

Los detalles sobre este paquete se encuentran en la Sección 8.29.2, “Contenido de GCC”.

Capítulo 7. Entrada al entorno chroot y creación de herramientas temporales adicionales

7.1. Introducción

Este capítulo muestra cómo crear los últimos elementos faltantes del sistema temporal: las herramientas necesarias para crear los distintos paquetes. Una vez resueltas todas las dependencias circulares, se puede utilizar un entorno chroot, completamente aislado del sistema operativo host (excepto del kernel en ejecución), para la creación.

Para el correcto funcionamiento del entorno aislado, se debe establecer comunicación con el kernel en ejecución. Esto se realiza mediante los denominados Sistemas de Archivos del Kernel Virtual, que se montarán antes de entrar en el entorno chroot. Puede verificar que estén montados ejecutando el comando **findmnt**.

Hasta la Sección 7.4, “Ingreso al entorno Chroot”, los comandos deben ejecutarse como root, con la variable LFS establecida. Después de ingresar al entorno Chroot, todos los comandos se ejecutan como root, afortunadamente sin acceso al sistema operativo del equipo en el que se creó LFS. Tenga cuidado, ya que es fácil destruir todo el sistema LFS con comandos erróneos.

7.2. Cambio de propiedad

Nota

Los comandos del resto de este libro deben ejecutarse con la sesión iniciada como root y ya no como lfs. Además, verifique que $LFS esté establecido en el entorno root.

Actualmente, toda la jerarquía de directorios en $LFS pertenece al usuario lfs, un usuario que solo existe en el sistema host.

Si los directorios y archivos de $LFS se mantienen como están, serán propiedad de un ID de usuario sin una cuenta correspondiente. Esto es peligroso, ya que una cuenta de usuario creada posteriormente podría obtener este mismo ID de usuario y ser propietaria de todos los archivos de $LFS, exponiéndolos así a una posible manipulación maliciosa. Para solucionar este problema, cambie la propiedad de los directorios $LFS/* al usuario root ejecutando el siguiente comando:

```
chown --from lfs -R root:root $LFS/{usr,lib,var,etc,bin,sbin,tools}
case $(uname -m) in
  x86_64) chown --from lfs -R root:root $LFS/lib64 ;;
esac
```

7.3. Preparación de los sistemas de archivos virtuales del kernel

Las aplicaciones que se ejecutan en el espacio de usuario utilizan varios sistemas de archivos creados por el kernel para comunicarse con él. Estos sistemas de archivos son virtuales: no se utiliza espacio de disco para ellos. El contenido de estos sistemas de archivos reside en la memoria. Estos sistemas de archivos deben montarse en el árbol de directorios $LFS para que las aplicaciones puedan encontrarlos en el entorno chroot. Comience creando los directorios donde se montarán estos sistemas de archivos virtuales:

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

7.3.1. Montaje y llenado de /dev

Durante un arranque normal de un sistema LFS, el kernel monta automáticamente el sistema de archivos devtmpfs en el directorio /dev.

El kernel crea nodos de dispositivo en ese sistema de archivos virtual durante el proceso de arranque o cuando se detecta o accede a un dispositivo por primera vez. El demonio udev puede cambiar la propiedad o los permisos de los nodos de dispositivo creados por el kernel, y crear nuevos nodos de dispositivo o enlaces simbólicos para facilitar el trabajo de los encargados del mantenimiento de la distribución y los administradores de sistemas. (Consulte la Sección 9.3.2.2, “Creación de Nodos de Dispositivo” para obtener más información). Si el kernel del host admite devtmpfs, podemos simplemente montar un devtmpfs en $LFS/dev y confiar en que el kernel lo rellene.

Sin embargo, algunos kernels del host no admiten devtmpfs; estas distribuciones utilizan métodos diferentes para crear el contenido de /dev. Por lo tanto, la única forma independiente del host de rellenar el directorio $LFS/dev es mediante un montaje de enlace del directorio /dev del sistema host. Un montaje de enlace es un tipo especial de montaje que hace visible un subárbol de directorios o un archivo en otra ubicación. Utilice el siguiente comando para ello:

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

7.3.2. Montaje de sistemas de archivos virtuales del kernel

Ahora monte los sistemas de archivos virtuales del kernel restantes:

```
mount -vt devpts devpts -o gid=5,mode=0620 $LFS/dev/pts
mount -vt proc proc $LFS/proc
mount -vt sysfs sysfs $LFS/sys
mount -vt tmpfs tmpfs $LFS/run
```

Significado de las opciones de montaje para devpts:

`gid=5`

Esto garantiza que todos los nodos de dispositivo creados por devpts pertenezcan al ID de grupo 5. Este es el ID que usaremos más adelante para el grupo tty. Usamos el ID de grupo en lugar de un nombre, ya que el sistema host podría usar un ID diferente para su grupo tty.

mode=0620

Esto garantiza que todos los nodos de dispositivo creados por devpts tengan el modo 0620 (lectura y escritura por parte del usuario, escritura por parte del grupo). Junto con la opción anterior, esto garantiza que devpts cree nodos de dispositivo que cumplan con los requisitos de grantpt(), lo que significa que el binario auxiliar pt_chown de Glibc (que no se instala por defecto) no es necesario.

En algunos sistemas host, /dev/shm es un enlace simbólico a un directorio, normalmente /run/shm. El archivo tmpfs /run se montó anteriormente, por lo que en este caso solo es necesario crear un directorio con los permisos correctos.

En otros sistemas host, /dev/shm es un punto de montaje para un archivo tmpfs. En ese caso, el montaje de /dev anterior solo creará /dev/shm como directorio en el entorno chroot. En esta situación, debemos montar explícitamente un archivo tmpfs:

```
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mount -vt tmpfs -o nosuid,nodev tmpfs $LFS/dev/shm
fi
```

7.4. Acceso al entorno chroot

Ahora que todos los paquetes necesarios para compilar el resto de las herramientas necesarias están en el sistema, es hora de acceder al entorno chroot y terminar de instalar las herramientas temporales. Este entorno también se utilizará para instalar el sistema final. Como usuario root, ejecute el siguiente comando para acceder al entorno que, actualmente, solo contiene herramientas temporales:

```
chroot "$LFS" /usr/bin/env -i   \
    HOME=/root                  \
    TERM="$TERM"                \
    PS1='(lfs chroot) \u:\w\$ ' \
    PATH=/usr/bin:/usr/sbin     \
    MAKEFLAGS="-j$(nproc)"      \
    TESTSUITEFLAGS="-j$(nproc)" \
    /bin/bash --login
```

Si no desea utilizar todos los núcleos lógicos disponibles, reemplace $(nproc) con el número de núcleos lógicos que desea utilizar para compilar paquetes en este capítulo y los siguientes. Las suites de pruebas de algunos paquetes (en particular, Autoconf, Libtool y Tar) del Capítulo 8 no se ven afectadas por MAKEFLAGS; en su lugar, utilizan la variable de entorno TESTSUITEFLAGS. También configuramos esto aquí para ejecutar estas suites de pruebas con múltiples núcleos.

La opción -i del comando **env** borrará todas las variables del entorno chroot. Después, solo se configuran de nuevo las variables HOME, TERM, PS1 y PATH. La construcción TERM=$TERM establece la variable TERM dentro del entorno chroot con el mismo valor que fuera del mismo. Esta variable es necesaria para que programas como vim y less funcionen correctamente. Si se desean otras variables, como CFLAGS o CXXFLAGS, este es un buen lugar para configurarlas.

A partir de este momento, ya no es necesario usar la variable LFS, ya que todo el trabajo se restringirá al sistema de archivos LFS; el comando **chroot** ejecuta la shell de Bash con el directorio raíz (/) establecido en $LFS.

Tenga en cuenta que /tools/bin no está en PATH. Esto significa que ya no se utilizará la cadena de herramientas cruzada.

Tenga en cuenta también que el prompt de **Bash** mostrará el mensaje "¡No tengo nombre!". Esto es normal, ya que el archivo /etc/passwd aún no se ha creado.

Nota

Es importante que todos los comandos de este capítulo y los siguientes se ejecuten desde el entorno chroot. Si abandona este entorno por cualquier motivo (por ejemplo, al reiniciar), asegúrese de que los sistemas de archivos virtuales del kernel estén montados como se explica en la Sección 7.3.1, “Montaje y llenado de /dev” y la Sección 7.3.2, “Montaje de sistemas de archivos virtuales del kernel” y vuelva a entrar en chroot antes de continuar con la instalación.

7.5. Creación de directorios

Es hora de crear la estructura completa de directorios en el sistema de archivos LFS.

Nota

Es posible que algunos de los directorios mencionados en esta sección ya se hayan creado previamente con instrucciones explícitas o al instalar algunos paquetes. Se repiten a continuación para mayor claridad. Cree algunos directorios de nivel raíz que no estén en el conjunto limitado requerido en los capítulos anteriores ejecutando el siguiente comando:

```
mkdir -pv /{boot,home,mnt,opt,srv}
```

Cree el conjunto de subdirectorios requerido por debajo del nivel raíz con los siguientes comandos:

```
mkdir -pv /etc/{opt,sysconfig}
mkdir -pv /lib/firmware
mkdir -pv /media/{floppy,cdrom}
mkdir -pv /usr/{,local/}{include,src}
mkdir -pv /usr/lib/locale
mkdir -pv /usr/local/{bin,lib,sbin}
mkdir -pv /usr/{,local/}share/{color,dict,doc,info,locale,man}
mkdir -pv /usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv /usr/{,local/}share/man/man{1..8}
mkdir -pv /var/{cache,local,log,mail,opt,spool}
mkdir -pv /var/lib/{color,misc,locate}

ln -sfv /run /var/run
ln -sfv /run/lock /var/lock

install -dv -m 0750 /root
install -dv -m 1777 /tmp /var/tmp
```

Los directorios se crean, por defecto, con el modo de permiso 755, pero esto no es recomendable en todos los casos. En los comandos anteriores, se realizan dos cambios: uno en el directorio de inicio del usuario root y otro en los directorios de archivos temporales.

El primer cambio de modo garantiza que no cualquiera pueda acceder al directorio /root, tal como lo haría un usuario normal con su propio directorio de inicio. El segundo cambio de modo garantiza que cualquier usuario pueda escribir en los directorios /tmp y /var/tmp, pero no puede eliminar los archivos de otro usuario. Esto último está prohibido por el llamado "bit pegajoso", el bit más alto (1) en la máscara de 1777 bits.

7.5.1. Nota de conformidad con FHS

Este árbol de directorios se basa en el Estándar de Jerarquía del Sistema de Archivos (FHS) (disponible en https://refspecs.linuxfoundation.org/fhs.shtml). El FHS también especifica la existencia opcional de directorios adicionales como /usr/local/games y /usr/share/games. En LFS, solo creamos los directorios realmente necesarios. Sin embargo, puede crear más directorios si lo desea.

Advertencia

El FHS no exige la existencia del directorio /usr/lib64, y los editores de LFS han decidido no usarlo. Para que las instrucciones de LFS y BLFS funcionen correctamente, es imperativo que este directorio no exista. Debe verificarlo de vez en cuando, ya que es fácil crearlo accidentalmente, lo que probablemente dañará su sistema.

7.6. Creación de archivos esenciales y enlaces simbólicos

Históricamente, Linux mantenía una lista de los sistemas de archivos montados en el archivo /etc/mtab. Los núcleos modernos mantienen esta lista internamente y la exponen al usuario a través del sistema de archivos /proc. Para satisfacer las necesidades de las utilidades que esperan encontrar /etc/mtab, cree el siguiente enlace simbólico:

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

Cree un archivo /etc/hosts básico que se referenciará en algunos conjuntos de pruebas y también en uno de los archivos de configuración de Perl:

```
cat > /etc/hosts << EOF
127.0.0.1  localhost $(nombre_del_host)
::1        localhost
EOF
```

Para que el usuario root pueda iniciar sesión y se reconozca el nombre "root", debe haber entradas relevantes en los archivos /etc/passwd y /etc/group. Cree el archivo /etc/passwd ejecutando el siguiente comando:

```
cat > /etc/passwd << "EOF"
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/dev/null:/usr/bin/false
daemon:x:6:6:Daemon User:/dev/null:/usr/bin/false
messagebus:x:18:18:D-Bus Message Daemon User:/run/dbus:/usr/bin/false
uuidd:x:80:80:UUID Generation Daemon User:/dev/null:/usr/bin/false
nobody:x:65534:65534:Unprivileged User:/dev/null:/usr/bin/false
EOF
```

La contraseña de root se establecerá más adelante. Cree el archivo /etc/group ejecutando el siguiente comando:

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

```
kvm:x:61:
uuidd:x:80:
wheel:x:97:
users:x:999:
nogroup:x:65534:
EOF
```

Los grupos creados no forman parte de ningún estándar; son grupos determinados en parte por los requisitos de la Configuración de Udev en el Capítulo 9, y en parte por las convenciones comunes empleadas por varias distribuciones de Linux existentes.

Además, algunas suites de pruebas dependen de usuarios o grupos específicos. La Base Estándar de Linux (LSB, disponible en *https://refspecs.linuxfoundation.org/lsb.shtml*) solo recomienda que, además del grupo root con un ID de Grupo (GID) de 0, exista un grupo bin con un GID de 1. El GID de 5 se usa ampliamente para el grupo tty, y el número 5 también se usa en /etc/fstab para el sistema de archivos devpts. El administrador del sistema puede elegir libremente todos los demás nombres de grupo e GID, ya que los programas bien escritos no dependen de los números de GID, sino que utilizan el nombre del grupo. El kernel utiliza el ID 65534 para NFS y espacios de nombres de usuario independientes para usuarios y grupos no asignados (estos existen en el servidor NFS o en el espacio de nombres de usuario principal, pero no existen en la máquina local ni en el espacio de nombres independiente).

Asignamos nobody y nogroup para evitar un ID sin nombre. Sin embargo, otras distribuciones pueden tratar este ID de forma diferente, por lo que ningún programa portátil debería depender de esta asignación.

Algunas pruebas del Capítulo 8 requieren un usuario regular. Añadimos este usuario aquí y eliminamos esta cuenta al final de ese capítulo.

```
echo "tester:x:101:101::/home/tester:/bin/bash" >> /etc/passwd
echo "tester:x:101:" >> /etc/group
install -o tester -d /home/tester
```

Para eliminar el mensaje "¡No tengo nombre!", inicie una nueva consola. Dado que los archivos /etc/passwd y /etc/group ya se han creado, la resolución de nombres de usuario y de grupo funcionará:

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

Los programas **login**, **agetty** e **init** (y otros) utilizan varios archivos de registro para registrar información como quién inició sesión en el sistema y cuándo. Sin embargo, estos programas no escribirán en los archivos de registro si no existen. Inicialice los archivos de registro y asígneles los permisos adecuados:

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

El archivo /var/log/wtmp registra todos los inicios y cierres de sesión. El archivo /var/log/lastlog registra la última vez que cada usuario inició sesión. El archivo /var/log/faillog registra los intentos fallidos de inicio de sesión. El archivo /var/log/btmp registra los intentos fallidos de inicio de sesión.

Nota

El archivo /run/utmp registra los usuarios que han iniciado sesión. Este archivo se crea dinámicamente en los scripts de arranque.

Nota

Los archivos utmp, wtmp, btmp y lastlog utilizan enteros de 32 bits para las marcas de tiempo y dejarán de funcionar completamente después del año 2038. Muchos paquetes han dejado de usarlos y otros dejarán de usarlos.
Probablemente sea mejor considerarlos obsoletos.

7.7. Gettext-0.24

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

Tiempo de compilación aproximado: 1.3 SBU
Espacio en disco requerido: 349 MB

7.7.1. Instalación de Gettext

Para nuestro conjunto temporal de herramientas, solo necesitamos instalar tres programas de Gettext. Preparar Gettext para la compilación:

```
./configure --disable-shared
```

Significado de la opción de configuración:

`--disable-shared`

No es necesario instalar ninguna de las bibliotecas compartidas de Gettext en este momento, por lo que no es necesario compilarlas.

Compilar el paquete:

```
make
```

Instalar los programas msgfmt, msgmerge y xgettext:

```
cp -v gettext-tools/src/{msgfmt,msgmerge,xgettext} /usr/bin
```

Los detalles de este paquete se encuentran en la Sección 8.33.2, “Contenido de Gettext”.

7.8. Bison-3.8.2

El paquete Bison contiene un generador de analizadores sintácticos.

Tiempo de compilación aproximado: 0,2 SBU
Espacio en disco requerido: 58 MB

7.8.1. Instalación de Bison

Preparar Bison para la compilación:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/bison-3.8.2
```

Significado de la nueva opción de configuración:

`--docdir=/usr/share/doc/bison-3.8.2`

Esto indica al sistema de compilación que instale la documentación de Bison en un directorio versionado.

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 8.34.2, “Contenido de Bison”.

7.9. Perl-5.40.1

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

Tiempo de compilación aproximado: 0,6 SBU
Espacio en disco requerido: 285 MB

7.9.1. Instalación de Perl

Preparar Perl para la compilación:

```
sh Configure -des                                         \
             -D prefix=/usr                               \
             -D vendorprefix=/usr                         \
             -D useshrplib                                \
             -D privlib=/usr/lib/perl5/5.40/core_perl     \
             -D archlib=/usr/lib/perl5/5.40/core_perl     \
             -D sitelib=/usr/lib/perl5/5.40/site_perl     \
             -D sitearch=/usr/lib/perl5/5.40/site_perl    \
             -D vendorlib=/usr/lib/perl5/5.40/vendor_perl \
             -D vendorarch=/usr/lib/perl5/5.40/vendor_perl
```

Significado de las opciones de configuración:

`-des`

Esta es una combinación de tres opciones: -d usa valores predeterminados para todos los elementos; -e garantiza la finalización de todas las tareas; -s silencia la salida no esencial.

`-D vendorprefix=/usr`

Esto garantiza que Perl sepa cómo indicar a los paquetes dónde deben instalar sus módulos Perl.

`-D useshrplib`

Construye libperl, necesaria para algunos módulos Perl, como una biblioteca compartida, en l ugar de una biblioteca estática.

`-D privlib, -D archlib, -D sitelib,...`

Estas opciones definen dónde Perl busca los módulos instalados. Los editores de LFS optaron por colocarlos en una estructura de directorios basada en la versión principal o secundaria de Perl (5.40), lo que permite actualizar Perl a versiones de parche más recientes (la versión de parche es la última parte separada por puntos en la cadena de la versión completa, como 5.40.1) sin reinstalar todos los módulos.

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 8.43.2, “Contenido de Perl”.

7.10. Python-3.13.2

El paquete Python 3 contiene el entorno de desarrollo de Python. Es útil para la programación orientada a objetos, la escritura de scripts, la creación de prototipos de programas grandes y el desarrollo de aplicaciones completas. Python es un lenguaje de programación interpretado.

Tiempo de compilación aproximado: 0,5 SBU
Espacio en disco requerido: 634 MB

7.10.1. Instalación de Python

Nota

Hay dos archivos de paquete cuyo nombre empieza con el prefijo "python". El que se debe extraer es Python3.13.2.tar.xz (observe la primera letra en mayúscula).

Preparar Python para la compilación:

```
./configure --prefix=/usr   \
            --enable-shared \
            --without-ensurepip
```

Significado de la opción de configuración:

`--enable-shared`

Esta opción impide la instalación de bibliotecas estáticas.

`--without-ensurepip`

Esta opción deshabilita el instalador de paquetes de Python, que no es necesario en esta etapa.

Compilar el paquete:

```
make
```

Nota

Algunos módulos de Python 3 no se pueden compilar ahora porque las dependencias aún no están instaladas. Para el módulo ssl, se muestra el mensaje "Python requiere OpenSSL 1.1.1 o posterior". Este mensaje debe ignorarse. Solo asegúrese de que el comando make de nivel superior no haya fallado. Los módulos opcionales no son necesarios ahora y se compilarán en el Capítulo 8.

Instalar el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 8.51.2, "Contenido de Python 3".

7.11. Texinfo-7.2

El paquete Texinfo contiene programas para leer, escribir y convertir páginas de información.

Tiempo de compilación aproximado: 0,2 SBU
Espacio en disco requerido: 152 MB

7.11.1. Instalación de Texinfo

Preparar Texinfo para la compilación:

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

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make install
```

Los detalles de este paquete se encuentran en la Sección 8.72.2, “Contenido de Texinfo”.

7.12. Util-linux-2.40.4

El paquete Util-linux contiene diversas utilidades.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 182 MB

7.12.1. Instalación de Util-linux

FHS recomienda usar el directorio /var/lib/hwclock en lugar del directorio /etc habitual como ubicación para el archivo adjtime. Cree este directorio con:

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

Prepare Util-linux para la compilación:

```
./configure --libdir=/usr/lib                      \
            --runstatedir=/run                     \
            --disable-chfn-chsh                    \
            --disable-login                        \
            --disable-nologin                      \
            --disable-su                           \
            --disable-setpriv                      \
            --disable-runuser                      \
            --disable-pylibmount                   \
            --disable-static                       \
            --disable-liblastlog2                  \
            --without-python                       \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.4
```

Significado de las opciones de configuración:

ADJTIME_PATH=/var/lib/hwclock/adjtime

Establece la ubicación del archivo que registra la información del reloj del hardware según el FHS. Esto no es estrictamente necesario para esta herramienta temporal, pero evita la creación de un archivo en otra ubicación, que no se sobrescribiría ni eliminaría al compilar el paquete final de util-linux.

--libdir=/usr/lib

Esta opción garantiza que los enlaces simbólicos .so apunten directamente al archivo de la biblioteca compartida en el mismo directorio (/usr/lib).

*--disable-**

Estas opciones evitan advertencias sobre la compilación de componentes que requieren paquetes que no están en LFS o que aún no están instalados.

--without-python

Esta opción deshabilita el uso de Python. Evita intentar compilar enlaces innecesarios.

runstatedir=/run

Esta opción establece correctamente la ubicación del socket utilizado por **uuidd** y libuuid.

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make install
```

Los detalles sobre este paquete se encuentran en la Sección 8.79.2, “Contenido de Util-linux”.

7.13. Limpieza y guardado del sistema temporal

7.13.1. Limpieza

Primero, elimine los archivos de documentación instalados para evitar que se instalen en el sistema final y ahorrar unos 35 MB:

```
rm -rf /usr/share/{info,man,doc}/*
```

Segundo, en un sistema Linux moderno, los archivos .la de libtool solo son útiles para libltdl. Libltdl no carga ninguna biblioteca en LFS, y se sabe que algunos archivos .la pueden causar fallos en los paquetes BLFS. Elimine esos archivos ahora:

```
find /usr/{lib,libexec} -name \*.la -delete
```

El tamaño actual del sistema es de unos 3 GB; sin embargo, el directorio /tools ya no es necesario. Ocupa aproximadamente 1 GB de espacio en disco. Elimínelo ahora:

```
rm -rf /tools
```

7.13.2. Copia de seguridad

En este punto, se han creado los programas y bibliotecas esenciales y su sistema LFS actual se encuentra en buen estado.

Ahora puede realizar una copia de seguridad de su sistema para su posterior reutilización. En caso de fallos fatales en los capítulos posteriores, suele resultar que eliminar todo y empezar de cero (con más cuidado) es la mejor manera de recuperarse. Desafortunadamente, también se eliminarán todos los archivos temporales. Para evitar perder tiempo rehaciendo algo que se ha hecho correctamente, puede resultar útil crear una copia de seguridad del sistema LFS actual.

Nota

Todos los pasos restantes de esta sección son opcionales. Sin embargo, en cuanto empiece a instalar paquetes en el capítulo 8, se sobrescribirán los archivos temporales. Por lo tanto, puede ser recomendable realizar una copia de seguridad del sistema actual como se describe a continuación.

Los siguientes pasos se realizan desde fuera del entorno chroot. Esto significa que primero debe salir del entorno chroot antes de continuar. El motivo es acceder a ubicaciones del sistema de archivos fuera del entorno chroot para almacenar/leer el archivo de copia de seguridad, que no debe estar dentro de la jerarquía $LFS.

Si ha decidido realizar una copia de seguridad, abandone el entorno chroot:

```
exit
```

Importante

Todas las siguientes instrucciones las ejecuta el usuario root en su sistema host. Tenga especial cuidado con los comandos que va a ejecutar, ya que los errores que cometa aquí pueden modificar su sistema host. Tenga en cuenta que la variable de entorno LFS está configurada para el usuario lfs por defecto, pero puede no estar configurada para el usuario root.
Siempre que el usuario root deba ejecutar comandos, asegúrese de haber configurado LFS.
Esto se ha explicado en la Sección 2.6, “Configuración de la variable $LFS y Umask”.

Antes de realizar una copia de seguridad, desmonte los sistemas de archivos virtuales:

```
mountpoint -q $LFS/dev/shm && umount $LFS/dev/shm
umount $LFS/dev/pts
umount $LFS/{sys,proc,run,dev}
```

Asegúrese de tener al menos 1 GB de espacio libre en disco (los archivos tar de origen se incluirán en el archivo de copia de seguridad) en el sistema de archivos que contiene el directorio donde creó el archivo de copia de seguridad.

Tenga en cuenta que las instrucciones a continuación especifican el directorio de inicio del usuario root del sistema host, que normalmente se encuentra en el sistema de archivos raíz. Reemplace $HOME por un directorio de su elección si no desea que la copia de seguridad se almacene en el directorio de inicio del usuario root.

Cree el archivo de copia de seguridad ejecutando el siguiente comando:

Nota

Debido a que el archivo de copia de seguridad está comprimido, tarda bastante tiempo (más de 10 minutos) incluso en un sistema relativamente rápido.

```
cd $LFS
tar -cJpf $HOME/lfs-temp-tools-12.3.tar.xz .
```

Nota

Si continúa con el capítulo 8, no olvide volver a acceder al entorno chroot como se explica en el recuadro "Importante" a continuación.

7.13.3. Restaurar

En caso de errores y necesite empezar de cero, puede usar esta copia de seguridad para restaurar el sistema y ahorrar tiempo de recuperación. Dado que las fuentes se encuentran en $LFS, también se incluyen en el archivo de copia de seguridad,

por lo que no es necesario volver a descargarlas. Después de comprobar que $LFS esté configurado correctamente, puede restaurar la copia de seguridad ejecutando los siguientes comandos:

Advertencia

Los siguientes comandos son extremadamente peligrosos. Si ejecuta rm -rf ./* como usuario root y no cambia al directorio $LFS o la variable de entorno LFS no está configurada para el usuario root, se destruirá todo el sistema host. ADVERTENCIA.

```
cd $LFS
rm -rf ./*
tar -xpf $HOME/lfs-temp-tools-12.3.tar.xz
```

De nuevo, verifique que el entorno se haya configurado correctamente y continúe compilando el resto del sistema.

Importante

Si abandonó el entorno chroot para crear una copia de seguridad o reiniciar la compilación mediante una restauración, recuerde comprobar que los sistemas de archivos virtuales sigan montados (findmnt | grep $LFS). Si no lo están, vuelva a montarlos como se describe en la Sección 7.3, “Preparación de los sistemas de archivos virtuales del kernel” y vuelva a acceder al entorno chroot (consulte la Sección 7.4, “Acceso al entorno chroot”) antes de continuar.

Parte IV. Compilación del sistema LFS

Capítulo 8. Instalación del software básico del sistema

8.1. Introducción

En este capítulo, comenzamos a construir el sistema LFS en serio.

La instalación de este software es sencilla. Aunque en muchos casos las instrucciones de instalación podrían ser más breves y genéricas, hemos optado por proporcionar las instrucciones completas de cada paquete para minimizar la posibilidad de errores. La clave para comprender el funcionamiento de un sistema Linux es saber para qué se utiliza cada paquete y por qué usted (o el sistema) podría necesitarlo.

No recomendamos el uso de optimizaciones personalizadas. Estas pueden acelerar un poco la ejecución del programa, pero también pueden causar dificultades de compilación y problemas al ejecutarlo. Si un paquete no compila con una optimización personalizada, intente compilarlo sin ella y compruebe si esto soluciona el problema. Incluso si el paquete compila con una optimización personalizada, existe el riesgo de que se haya compilado incorrectamente debido a las complejas interacciones entre el código y las herramientas de compilación. Tenga en cuenta también que las opciones `-march` y `-mtune` que usan valores no especificados en el manual no se han probado. Esto podría causar problemas con los paquetes de la cadena de herramientas (Binutils, GCC y Glibc). Las pequeñas ventajas potenciales que se obtienen al personalizar las optimizaciones del compilador a menudo se ven superadas por los riesgos. Se recomienda a quienes crean LFS por primera vez que lo hagan sin optimizaciones personalizadas.

Por otro lado, mantenemos las optimizaciones habilitadas por la configuración predeterminada de los paquetes. Además, en ocasiones habilitamos explícitamente una configuración optimizada proporcionada por un paquete, pero que no está habilitada por defecto. Los responsables del paquete ya han probado estas configuraciones y las consideran seguras, por lo que es improbable que interrumpan la compilación. Generalmente, la configuración predeterminada ya habilita `-O2` u `-O3`, por lo que el sistema resultante seguirá funcionando muy rápido sin ninguna optimización personalizada y será estable al mismo tiempo. Antes de las instrucciones de instalación, cada página proporciona información sobre el paquete, incluyendo una descripción concisa de su contenido, el tiempo aproximado de compilación y el espacio en disco requerido durante el proceso. Tras las instrucciones de instalación, se incluye una lista de programas y bibliotecas (junto con breves descripciones) que instala el paquete.

Nota

Los valores de SBU y el espacio en disco requerido incluyen datos de la suite de pruebas para todos los paquetes aplicables en el Capítulo 8. Los valores de SBU se han calculado utilizando cuatro núcleos de CPU (-j4) para todas las operaciones, a menos que se especifique lo contrario.

8.1.1. Acerca de las bibliotecas

En general, los editores de LFS desaconsejan compilar e instalar bibliotecas estáticas. La mayoría de las bibliotecas estáticas se han vuelto obsoletas en un sistema Linux moderno. Además, vincular una

biblioteca estática a un programa puede ser perjudicial. Si se necesita una actualización de la biblioteca para solucionar un problema de seguridad, todos los programas que la utilicen deberán volver a vincularse con la nueva biblioteca. Dado que el uso de bibliotecas estáticas no siempre es obvio, es posible que ni siquiera se conozcan los programas relevantes (ni los procedimientos necesarios para realizar la vinculación).

Los procedimientos de este capítulo eliminan o deshabilitan la instalación de la mayoría de las bibliotecas estáticas. Normalmente, esto se realiza mediante la opción `--disable-static` para la **configuración**. En otros casos, se requieren métodos alternativos. En algunos casos, especialmente en Glibc y GCC, el uso de bibliotecas estáticas sigue siendo una característica esencial del proceso de creación de paquetes.

Para una explicación más completa sobre las bibliotecas, véase *Bibliotecas: ¿Estáticas o compartidas?* en el libro BLFS.

8.2. Gestión de Paquetes

La gestión de paquetes es una adición frecuente al libro de LFS. Un gestor de paquetes monitoriza la instalación de archivos, lo que facilita la eliminación y actualización de paquetes. Un buen gestor de paquetes también gestionará los archivos de configuración, especialmente para mantener la configuración del usuario al reinstalar o actualizar el paquete. Antes de que empiece a dudar, NO: esta sección no abordará ni recomendará ningún gestor de paquetes en particular. Lo que sí ofrece es un resumen de las técnicas más populares y su funcionamiento. El gestor de paquetes perfecto para usted puede estar entre estas técnicas, o puede ser una combinación de dos o más. Esta sección menciona brevemente los problemas que pueden surgir al actualizar paquetes.

Algunas razones por las que no se menciona ningún gestor de paquetes en LFS ni en BLFS incluyen:

• Tratar con la gestión de paquetes desvía la atención del objetivo de estos libros: enseñar cómo se construye un sistema Linux.

• Existen múltiples soluciones para la gestión de paquetes, cada una con sus ventajas y desventajas. Encontrar una solución que satisfaga a todos los públicos es difícil. Hay algunos consejos sobre la gestión de paquetes. Visite el Proyecto Hints y compruebe si alguno se ajusta a sus necesidades.

8.2.1. Problemas de actualización

Un gestor de paquetes facilita la actualización a versiones más recientes cuando se publican. Generalmente, se pueden usar las instrucciones de los libros LFS y BLFS para actualizar a las versiones más recientes. Aquí hay algunos puntos que debe tener en cuenta al actualizar paquetes, especialmente en un sistema en ejecución.

• Si es necesario actualizar el kernel de Linux (por ejemplo, de 5.10.17 a 5.10.18 o 5.11.1), no es necesario reconstruir nada más. El sistema seguirá funcionando correctamente gracias a la interfaz bien definida entre el kernel y el espacio de usuario.

En concreto, no es necesario actualizar las cabeceras de la API de Linux junto con el kernel. Simplemente tendrá que reiniciar el sistema para usar el kernel actualizado.

• Si es necesario actualizar Glibc a una versión más reciente (por ejemplo, de Glibc-2.36 a Glibc-2.41), se requieren pasos adicionales para evitar fallos en el sistema. Consulte la Sección 8.5, “Glibc-2.41” para obtener más información.

• Si se actualiza un paquete que contiene una biblioteca compartida y el nombre de la biblioteca cambia, cualquier paquete enlazado dinámicamente a la biblioteca debe recompilarse para enlazarlo con la biblioteca más reciente. (Tenga en cuenta que no existe correlación entre la versión del paquete y el nombre de la biblioteca). Por ejemplo, considere un paquete foo-1.2.3 que instala una biblioteca compartida llamada libfoo.so.1. Suponga que actualiza el paquete a una versión más reciente, foo-1.2.4, que instala una biblioteca compartida llamada libfoo.so.2. En este caso, cualquier paquete enlazado dinámicamente a libfoo.so.1 debe recompilarse para enlazar con libfoo.so.2 y usar la nueva versión de la biblioteca.

No debe eliminar las bibliotecas antiguas hasta que se hayan recompilado todos los paquetes dependientes.

• Si un paquete está enlazado (directa o indirectamente) tanto al nombre antiguo como al nuevo de una biblioteca compartida (por ejemplo, el paquete enlaza tanto a libfoo.so.2 como a libbar.so.1, mientras que este último enlaza a libfoo.so.3), el paquete podría funcionar mal debido a que las diferentes revisiones de la biblioteca compartida presentan definiciones incompatibles para algunos nombres de símbolos. Esto puede deberse a la recompilación de algunos, pero no todos, los paquetes enlazados a la antigua biblioteca compartida después de actualizar el paquete que la proporciona. Para evitar el problema, los usuarios deberán reconstruir cada paquete vinculado a una biblioteca compartida con una revisión actualizada (p. ej., libfoo.so.2 a libfoo.so.3) lo antes posible.

• Si se actualiza un paquete que contiene una biblioteca compartida y el nombre de la biblioteca no cambia, pero el número de versión del archivo de la biblioteca disminuye (por ejemplo, la biblioteca sigue llamándose libfoo.so.1, pero el nombre del archivo de la biblioteca cambia de libfoo.so.1.25 a libfoo.so.1.24), debe eliminar el archivo de la biblioteca de la versión instalada previamente (libfoo.so.1.25 en este caso). De lo contrario, un comando ldconfig (invocado por usted mismo desde la línea de comandos o al instalar algún paquete) restablecerá el enlace simbólico libfoo.so.1 para que apunte al archivo de biblioteca anterior, ya que parece ser una versión más reciente; su número de versión es mayor. Esta situación puede surgir si tiene que degradar un paquete o si los autores cambian el esquema de versiones de los archivos de la biblioteca.

• Si se actualiza un paquete que contiene una biblioteca compartida y el nombre de la biblioteca no cambia, pero se soluciona un problema grave (en particular, una vulnerabilidad de seguridad), se deben reiniciar todos los programas en ejecución vinculados a la biblioteca compartida.

El siguiente comando, ejecutado como root una vez completada la actualización, mostrará qué procesos utilizan las versiones anteriores de esas bibliotecas (reemplace libfoo con el nombre de la biblioteca):

```
grep -l 'libfoo.*deleted' /proc/*/maps | tr -cd 0-9\\n | xargs -r ps u
```

Si se utiliza OpenSSH para acceder al sistema y este está vinculado a la biblioteca actualizada, debe reiniciar el servicio sshd, cerrar sesión, volver a iniciarla y ejecutar el comando anterior para confirmar que no haya ningún programa utilizando las bibliotecas eliminadas.

• Si se sobrescribe un programa ejecutable o una biblioteca compartida, los procesos que utilizan el código o los datos de ese programa o biblioteca podrían bloquearse. La forma correcta de actualizar un programa o una biblioteca compartida sin que el proceso se bloquee es eliminarlo primero y luego instalar la nueva versión. El comando de **instalación** de coreutils ya implementa esto, y la mayoría de los paquetes lo usan para instalar archivos binarios y bibliotecas. Esto significa que no tendrá problemas con este problema la mayor parte del tiempo. Sin embargo, el proceso de instalación de algunos paquetes (en particular, SpiderMonkey en BLFS) simplemente sobrescribe el archivo si existe; esto provoca un bloqueo. Por lo tanto, es más seguro guardar el trabajo y cerrar los procesos en ejecución innecesarios antes de actualizar un paquete.

8.2.2. Técnicas de gestión de paquetes

A continuación, se presentan algunas técnicas comunes de gestión de paquetes. Antes de decidirse por un gestor de paquetes, investigue las distintas técnicas, especialmente las desventajas de cada esquema.

8.2.2.1. ¡Todo está en mi cabeza!

Sí, esta es una técnica de gestión de paquetes. Algunas personas no necesitan un gestor de paquetes porque conocen los paquetes a la perfección y saben qué archivos instala cada uno. Otros usuarios tampoco necesitan ninguna gestión de paquetes porque planean reconstruir todo el sistema cada vez que se modifica un paquete.

8.2.2.2. Instalar en directorios separados

Esta es una técnica sencilla de gestión de paquetes que no requiere un programa especial para administrar los paquetes. Cada paquete se instala en un directorio separado. Por ejemplo, el paquete foo-1.1 se instala en `/opt/foo-1.1` y se crea un enlace simbólico de `/opt/foo a /opt/foo-1.1`. Cuando se publica una nueva versión de foo-1.2, se instala en `/opt/foo-1.2` y el enlace simbólico anterior se reemplaza por uno a la nueva versión.

Las variables de entorno como `PATH, MANPATH, INFOPATH, PKG_CONFIG_PATH, CPPFLAGS, LDFLAGS` y el archivo de configuración `/etc/ld.so.conf` podrían necesitar expandirse para incluir los subdirectorios correspondientes en `/opt/foo-x.y`.

El manual de BLFS utiliza este esquema para instalar algunos paquetes muy grandes y facilitar su actualización. Si se instalan varios paquetes, este esquema se vuelve inmanejable. Además, algunos paquetes (por ejemplo, las cabeceras de la API de Linux y Glibc) podrían no funcionar correctamente con este esquema. **Nunca lo utilice en todo el sistema**.

8.2.2.3. Gestión de paquetes con enlaces simbólicos

Esta es una variación de la técnica anterior de gestión de paquetes. Cada paquete se instala como en el esquema anterior.

Pero en lugar de crear el enlace simbólico mediante un nombre de paquete genérico, cada archivo se enlaza simbólicamente a la jerarquía `/usr`. Esto elimina la necesidad de expandir las variables de entorno. Aunque el usuario puede crear los enlaces simbólicos, muchos gestores de paquetes utilizan este enfoque y automatizan su creación. Algunos de los más populares son Stow, Epkg, Graft y Depot.

Es necesario engañar al script de instalación para que el paquete crea que está instalado en `/usr`, aunque en realidad esté instalado en la jerarquía `/usr/pkg`. Instalar de esta manera no suele ser una tarea sencilla. Por ejemplo, supongamos que se instala el paquete libfoo-1.1. Es posible que las siguientes instrucciones no instalen el paquete correctamente:

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

La instalación funcionará, pero es posible que los paquetes dependientes no se enlacen a libfoo como cabría esperar. Si compila un paquete que enlaza con libfoo, podría observar que se enlaza a `/usr/pkg/libfoo/1.1/lib/libfoo.so.1` en lugar de `/usr/lib/libfoo.so.1`, como cabría esperar. El enfoque correcto es usar la variable `DESTDIR` para dirigir la instalación.

Este enfoque funciona de la siguiente manera:

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

La mayoría de los paquetes admiten este enfoque, pero algunos no. Para los paquetes no compatibles, es posible que deba instalarlos manualmente o que le resulte más fácil instalar algunos paquetes problemáticos en `/opt`.

8.2.2.4. Basado en marca de tiempo

En esta técnica, se marca la hora de un archivo antes de la instalación del paquete. Tras la instalación, basta con usar el comando **find** con las opciones adecuadas para generar un registro de todos los archivos instalados tras la creación del archivo de marca de tiempo. Un gestor de paquetes que utiliza este enfoque es install-log.

Aunque este esquema tiene la ventaja de ser sencillo, presenta dos inconvenientes. Si, durante la instalación, los archivos se instalan con una marca de tiempo distinta a la actual, el gestor de paquetes no los rastreará. Además, este esquema solo se puede utilizar cuando los paquetes se instalan uno a la vez. Los registros no son fiables si se instalan dos paquetes simultáneamente desde dos consolas diferentes.

8.2.2.5. Seguimiento de scripts de instalación

En este enfoque, se registran los comandos que ejecutan los scripts de instalación. Existen dos técnicas que se pueden utilizar:

La variable de entorno `LD_PRELOAD` se puede configurar para que apunte a una biblioteca que se precargue antes de la instalación. Durante la instalación, esta biblioteca rastrea los paquetes que se instalan adjuntándose a varios ejecutables como **cp**, **install** y **mv**, y rastreando las llamadas al sistema que modifican el sistema de archivos. Para que este método funcione, todos los ejecutables deben estar enlazados dinámicamente sin el bit suid ni sgid. Precargar la biblioteca puede causar efectos secundarios no deseados durante la instalación. Por lo tanto, conviene realizar pruebas para garantizar que el gestor de paquetes no produzca errores y que registre todos los archivos correspondientes. Otra técnica es usar **strace**, que registra todas las llamadas al sistema realizadas durante la ejecución de los scripts de instalación.

8.2.2.6. Creación de Archivos de Paquetes

En este esquema, la instalación del paquete se simula en un árbol separado, como se describió previamente en la sección de administración de paquetes con enlaces simbólicos. Tras la instalación, se

crea un archivo de paquetes con los archivos instalados. Este archivo se utiliza para instalar el paquete en el equipo local o incluso en otros equipos.

Este enfoque lo utilizan la mayoría de los gestores de paquetes de las distribuciones comerciales. Ejemplos de gestores de paquetes que siguen este enfoque son RPM (que, por cierto, es requerido por la *Especificación Base del Estándar de Linux o LSB*), pkg-utils, apt de Debian y el sistema Portage de Gentoo. Encontrará una sugerencia sobre cómo adoptar este estilo de administración de paquetes para sistemas LFS en:

https://www.linuxfromscratch.org/hints/downloads/files/fakeroot.txt

La creación de archivos de paquetes que incluyen información de dependencias es compleja y queda fuera del alcance de LFS.

Slackware utiliza un sistema basado en tar para los archivos de paquetes. Este sistema no gestiona las dependencias de los paquetes a propósito como lo hacen los gestores de paquetes más complejos. Para más detalles sobre la gestión de paquetes de Slackware, consulte:

https://www.slackbook.org/html/package-management.html

8.2.2.7. Gestión basada en usuarios

Este esquema, exclusivo de LFS, fue ideado por Matthias Benkmann y está disponible en el Proyecto Hints. En este esquema, cada paquete se instala como un usuario independiente en las ubicaciones estándar. Los archivos pertenecientes a un paquete se identifican fácilmente comprobando el ID de usuario. Las características y desventajas de este enfoque son demasiado complejas para describirlas en esta sección. Para más detalles, consulte la sugerencia en:

https://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt

8.2.3. Implementación de LFS en varios sistemas

Una de las ventajas de un sistema LFS es que no hay archivos que dependan de su ubicación en un sistema de discos. Clonar una compilación LFS en otro equipo con la misma arquitectura que el sistema base es tan sencillo como usar tar en la partición LFS que contiene el directorio raíz (unos 900 MB sin comprimir para una compilación LFS básica), copiar ese archivo mediante transferencia de red o CD-ROM/memoria USB al nuevo sistema y expandirlo. Después, será necesario modificar algunos archivos de configuración. Los archivos de configuración que podrían necesitar actualizarse incluyen: /etc/hosts, /etc/fstab, /etc/passwd, /etc/group, /etc/shadow, /etc/ld.so.conf, /etc/sysconfig/rc.site, /etc/sysconfig/network y /etc/sysconfig/ifconfig.eth0.
Es posible que se necesite un kernel personalizado para el nuevo sistema, dependiendo de las diferencias en el hardware del sistema y la configuración original del kernel.

Nota

Se han reportado algunos problemas al copiar entre arquitecturas similares, pero no idénticas. Por ejemplo, el conjunto de instrucciones de un sistema Intel no es idéntico al de un procesador AMD, y las versiones posteriores de algunos procesadores pueden proporcionar instrucciones que no están disponibles en versiones anteriores.

Finalmente, el nuevo sistema debe ser arrancable mediante la Sección 10.4, "Uso de GRUB para configurar el proceso de arranque".

8.3. Páginas de manual - 6.12

El paquete de páginas de manual contiene más de 2400 páginas de manual.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 52 MB

8.3.1. Instalación de páginas de manual

Elimine dos páginas de manual para las funciones de hash de contraseñas. Libxcrypt proporcionará una versión mejorada de estas páginas de manual:

```
rm -v man3/crypt*
```

Instale las páginas de manual ejecutando:

```
make -R GIT=false prefix=/usr install
```

Significado de las opciones:

-R

Esto impide que **make** configure variables integradas. El sistema de compilación de páginas de manual no funciona bien con variables integradas, pero actualmente no hay forma de desactivarlas, excepto pasando explícitamente `-R` vía línea de comandos.

GIT=false

Esto evita que el sistema de compilación emita muchas líneas de advertencia «`git: command not found`».

8.3.2. Contenido de las páginas de manual

Archivos instalados: Varias páginas de manual

Descripciones breves

`man pages`	Descripción de las funciones del lenguaje de programación C, archivos importantes de dispositivos y archivos de configuración significativos

8.4. Iana-Etc-20250123

El paquete Iana-Etc proporciona datos para servicios y protocolos de red.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 4,8 MB

8.4.1. Instalación de Iana-Etc

Para este paquete, solo necesitamos copiar los archivos:

```
cp services protocols /etc
```

8.4.2. Contenido de Iana-Etc

Archivos instalados: /etc/protocols y /etc/services

Descripciones breves

`/etc/protocols`	Describe los diversos protocolos de Internet de DARPA disponibles en el subsistema TCP/IP
`/etc/services`	Proporciona una correspondencia entre los nombres textuales descriptivos de los servicios de Internet y sus números de puerto y tipos de protocolo asignados subyacentes

8.5. Glibc-2.41

El paquete Glibc contiene la biblioteca principal de C. Esta biblioteca proporciona las rutinas básicas para asignar memoria, buscar directorios, abrir y cerrar archivos, leer y escribir archivos, manejar cadenas, comparar patrones, realizar operaciones aritméticas, etc.

Tiempo de compilación aproximado: 12 SBU
Espacio en disco requerido: 3,2 GB

8.5.1. Instalación de Glibc

Algunos programas de Glibc utilizan el directorio `/var/db`, que no cumple con FHS, para almacenar sus datos de ejecución. Aplique el siguiente parche para que dichos programas almacenen sus datos de ejecución en ubicaciones compatibles con FHS:

```
patch -Np1 -i ../glibc-2.41-fhs-1.patch
```

La documentación de Glibc recomienda compilar Glibc en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Asegúrese de que las utilidades **ldconfig** y **sln** se instalen en `/usr/sbin`:

```
echo "rootsbindir=/usr/sbin" > configparms
```

Prepare Glibc para la compilación:

```
../configure --prefix=/usr                      \
             --disable-werror                    \
             --enable-kernel=5.4                \
             --enable-stack-protector=strong \
             --disable-nscd                     \
             libc_cv_slibdir=/usr/lib
```

Significado de las opciones de configuración:

`--disable-werror`

Esta opción deshabilita la opción -Werror pasada a GCC. Esto es necesario para ejecutar el conjunto de pruebas.

`--enable-kernel=5.4`

Esta opción indica al sistema de compilación que esta Glibc puede usarse con kernels de hasta la versión 5.4. Esto implica generar soluciones alternativas en caso de que no se pueda usar una llamada al sistema introducida en una versión posterior.

`--enable-stack-protector=strong`

Esta opción aumenta la seguridad del sistema al añadir código adicional para detectar desbordamientos de búfer, como ataques de destrucción de pila. Tenga en cuenta que Glibc siempre anula explícitamente el valor predeterminado de GCC, por lo que esta opción sigue siendo necesaria aunque ya hayamos especificado *`--enable-default-ssp`* para GCC.

`--disable-nscd`

No se compila el demonio de caché del servicio de nombres que ya no se utiliza.

`libc_cv_slibdir=/usr/lib`

Esta variable establece la biblioteca correcta para todos los sistemas. No queremos que se use lib64.

Compilar el paquete:

```
make
```

Importante

`En esta sección, el conjunto de pruebas de Glibc se considera crítico. No lo omita bajo ninguna circunstancia.`

Generalmente, algunas pruebas no superan los requisitos. Los fallos de prueba que se enumeran a continuación se pueden ignorar.

```
make check
```

Es posible que observe algunos fallos de prueba. El conjunto de pruebas de Glibc depende en cierta medida del sistema host. De entre más de 6000 pruebas, algunos fallos generalmente se pueden ignorar. Esta es una lista de los problemas más comunes observados en las versiones recientes de LFS:

• Se sabe que *io/tst-lchmod* falla en el entorno chroot de LFS.

• Se sabe que algunas pruebas, por ejemplo, `nss/tst-nss-files-hosts-multi` y `nptl/tst-thread-affinity*`, fallan debido a un tiempo de espera (especialmente cuando el sistema es relativamente lento o se ejecuta el conjunto de pruebas con varios trabajos de make en paralelo). Estas pruebas se pueden identificar con:

```
grep "Timed out" $(find -name \*.out)
```

Es posible volver a ejecutar una sola prueba con un tiempo de espera mayor con **TIMEOUTFACTOR=<*factor*> make test t=<*test name*>**. Por ejemplo, **TIMEOUTFACTOR=10 make test t=nss/tst-nss-files-hosts-multi** volverá a ejecutar *nss/tst-nss-files-hosts-multi* con un tiempo de espera diez veces mayor que el original.

• Además, algunas pruebas pueden fallar con un modelo de CPU relativamente antiguo (por ejemplo, *elf/tst-cpu-features-cpuinfo*) o una versión del kernel del host (por ejemplo, s*tdlib/tst-arc4random-thread*).

Aunque es un mensaje inofensivo, la instalación de Glibc mostrará un mensaje sobre la ausencia de `/etc/ld.so.conf`. Evitar esta advertencia con:

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

Corregir el Makefile para que omita una comprobación de seguridad obsoleta que falla con una configuración moderna de Glibc:

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

Importante

Si actualiza Glibc a una nueva versión menor (por ejemplo, de Glibc-2.36 a Glibc-2.41) en un sistema LFS en ejecución, debe tomar precauciones adicionales para evitar dañar el sistema:
• No se admite la actualización de Glibc en un sistema LFS anterior a la versión 11.0 (exclusiva). Reconstruya LFS si está ejecutando un sistema LFS tan antiguo como este, pero necesita una Glibc más reciente.
• Si actualiza en un sistema LFS anterior a la versión 12.0 (exclusiva), instale Libxcrypt siguiendo la Sección 8.27, “Libxcrypt-4.4.38”. Además de la instalación normal de Libxcrypt, **DEBE seguir la nota en la sección Libxcrypt para instalar libcrypt.so.1* (que reemplaza libcrypt.so.1 de la instalación anterior de Glibc)**.
• Si actualiza en un sistema LFS anterior a la versión 12.1 (exclusiva), elimine el programa **nscd**:

```
rm -f /usr/sbin/nscd
```

Actualice el kernel y reinicie si es anterior a la versión 5.4 (verifique la versión actual con **uname -r**) o si desea actualizarlo de todas formas, siga la Sección 10.3, “Linux-6.13.4”.
Actualice las cabeceras de la API del kernel si es anterior a la versión 5.4 (verifique la versión actual con **cat /usr/include/linux/version.h**) o si desea actualizarlo de todas formas, siga la Sección 5.4, “Cabeceras de la API de Linux-6.13.4” (pero elimine $LFS del comando **cp**). Realice una instalación de DESTDIR y actualice las bibliotecas compartidas de Glibc en el sistema con un solo comando de **instalación**:

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/*.so.* /usr/lib
```

Es fundamental seguir estrictamente estos pasos a menos que comprenda completamente lo que está haciendo. **Cualquier desviación inesperada puede inutilizar el sistema por completo. ADVERTENCIA**.
Luego, continúe ejecutando el comando **make install**, el comando sed en /usr/bin/ldd y los comandos para instalar las configuraciones regionales. Una vez finalizado, reinicie el sistema inmediatamente. Tras reiniciar el sistema correctamente, si está ejecutando un sistema LFS anterior a la versión 12.0 (exclusiva) donde GCC no se creó con la opción --disable-fixincludes, mueva dos encabezados de GCC a una mejor ubicación y elimine las copias obsoletas "corregidas" de los encabezados de Glibc:

```
DIR=$(dirname $(gcc -print-libgcc-file-name))
[ -e $DIR/include/limits.h ] || mv $DIR/include{-fixed,}/limits.h
[ -e $DIR/include/syslimits.h ] || mv $DIR/include{-fixed,}/syslimits.h
rm -rfv $(dirname $(gcc -print-libgcc-file-name))/include-fixed/*
```

Instalar el paquete:

```
make install
```

Corregir una ruta codificada al cargador de ejecutables en el script ldd:

```
sed '/RTLDLIST=/s@/usr@@g' -i /usr/bin/ldd
```

A continuación, instalar las configuraciones regionales que permiten que el sistema responda en un idioma diferente. Ninguna de estas configuraciones regionales es necesaria, pero si falta alguna, las suites de pruebas de algunos paquetes omitirán casos de prueba importantes.

Se pueden instalar configuraciones regionales individuales mediante el programa **localedef**. Por ejemplo, el segundo comando **localedef** a continuación combina la definición de configuración regional independiente del juego de caracteres `/usr/share/i18n/locales/cs_CZ` con la definición de mapa de caracteres `/usr/share/i18n/charmaps/UTF-8.gz` y añade el resultado al archivo `/usr/lib/locale/locale-archive`. Las siguientes instrucciones instalarán el conjunto mínimo de configuraciones regionales necesario para una cobertura óptima de las pruebas:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i cs_CZ -f UTF-8 cs_CZ.UTF-8
localedef -i de_DE -f ISO-8859-1 de_DE
localedef -i de_DE@euro -f ISO-8859-15 de_DE@euro
localedef -i de_DE -f UTF-8 de_DE.UTF-8
localedef -i el_GR -f ISO-8859-7 el_GR
localedef -i en_GB -f ISO-8859-1 en_GB
localedef -i en_GB -f UTF-8 en_GB.UTF-8
localedef -i en_HK -f ISO-8859-1 en_HK
localedef -i en_PH -f ISO-8859-1 en_PH
localedef -i en_US -f ISO-8859-1 en_US
localedef -i en_US -f UTF-8 en_US.UTF-8
localedef -i es_ES -f ISO-8859-15 es_ES@euro
localedef -i es_MX -f ISO-8859-1 es_MX
localedef -i fa_IR -f UTF-8 fa_IR
localedef -i fr_FR -f ISO-8859-1 fr_FR
localedef -i fr_FR@euro -f ISO-8859-15 fr_FR@euro
localedef -i fr_FR -f UTF-8 fr_FR.UTF-8
localedef -i is_IS -f ISO-8859-1 is_IS
localedef -i is_IS -f UTF-8 is_IS.UTF-8
localedef -i it_IT -f ISO-8859-1 it_IT
localedef -i it_IT -f ISO-8859-15 it_IT@euro
localedef -i it_IT -f UTF-8 it_IT.UTF-8
localedef -i ja_JP -f EUC-JP ja_JP
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
localedef -i nl_NL@euro -f ISO-8859-15 nl_NL@euro
localedef -i ru_RU -f KOI8-R ru_RU.KOI8-R
localedef -i ru_RU -f UTF-8 ru_RU.UTF-8
localedef -i se_NO -f UTF-8 se_NO.UTF-8
localedef -i ta_IN -f UTF-8 ta_IN.UTF-8
localedef -i tr_TR -f UTF-8 tr_TR.UTF-8
localedef -i zh_CN -f GB18030 zh_CN.GB18030
localedef -i zh_HK -f BIG5-HKSCS zh_HK.BIG5-HKSCS
localedef -i zh_TW -f UTF-8 zh_TW.UTF-8
```

Además, instale la configuración regional para su país, idioma y conjunto de caracteres.

Como alternativa, instale todas las configuraciones regionales listadas en el archivo `glibc-2.41/localedata/SUPPORTED` (que incluye todas las configuraciones regionales mencionadas anteriormente y muchas más) a la vez con el siguiente comando, que requiere mucho tiempo:

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

A continuación, utilice el comando localedef para crear e instalar las configuraciones regionales que no aparecen en el archivo glibc-2.41/localedata/SUPPORTED

cuando las necesite. Por ejemplo, las dos configuraciones regionales siguientes son necesarias para algunas pruebas posteriores en este capítulo:

```
localedef -i C -f UTF-8 C.UTF-8
localedef -i ja_JP -f SHIFT_JIS ja_JP.SJIS 2> /dev/null || true
```

Nota

Glibc ahora utiliza libidn2 al resolver nombres de dominio internacionalizados. Esta es una dependencia en tiempo de ejecución. Si se necesita esta capacidad, las instrucciones para instalar libidn2 se encuentran en la página de *libidn2* de *BLFS*.

8.5.2. Configuración de Glibc

8.5.2.1. Adición de nsswitch.conf

Es necesario crear el archivo *`/etc/nsswitch.conf`*, ya que la configuración predeterminada de Glibc no funciona correctamente en un entorno de red.

Cree un nuevo archivo *`/etc/nsswitch.conf`* ejecutando lo siguiente:

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

passwd: files
group: files
shadow: files

hosts: files dns
networks: files

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

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

8.5.2.2. Añadir datos de zona horaria

Instale y configure los datos de zona horaria con lo siguiente:

```
tar -xf ../../tzdata2025a.tar.gz

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

for tz in etcetera southamerica northamerica europe africa antarctica \
          asia australasia backward; do
    zic -L /dev/null    -d $ZONEINFO       ${tz}
    zic -L /dev/null    -d $ZONEINFO/posix ${tz}
    zic -L leapseconds  -d $ZONEINFO/right ${tz}
done

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

Significado de los comandos zic:

`zic -L /dev/null ...`

Esto crea zonas horarias POSIX sin segundos intercalares. Normalmente, se incluyen tanto en zoneinfo como en zoneinfo/posix. Es necesario incluir las zonas horarias POSIX en zoneinfo; de lo contrario, varias suites de pruebas informarán errores. En un sistema embebido, donde el espacio es limitado y no se pretende actualizar nunca las zonas horarias, se podrían ahorrar 1,9

MB si no se utiliza el directorio posix, pero algunas aplicaciones o conjuntos de pruebas podrían producir fallos.

`zic -L leapseconds ...`

Esto crea zonas horarias correctas, incluyendo segundos intercalares. En un sistema embebido, donde el espacio es limitado y no se pretende actualizar nunca las zonas horarias ni preocuparse por la hora correcta, se podrían ahorrar 1,9 MB si se omite el directorio correcto.

`zic ... -p ...`

Esto crea el archivo `posixrules`. Usamos Nueva York porque POSIX requiere que las reglas del horario de verano se ajusten a las de EE. UU.

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

```
tzselect
```

Tras responder algunas preguntas sobre la ubicación, el script mostrará el nombre de la zona horaria (p. ej., *América/Edmonton*). También hay otras zonas horarias posibles en `/usr/share/zoneinfo`, como Canadá/Este o EST5EDT, que no están identificadas por el script, pero que se pueden usar.

A continuación, cree el archivo `/etc/localtime` ejecutando:

```
ln -sfv /usr/share/zoneinfo/<xxx> /etc/localtime
```

Reemplace <xxx> con el nombre de la zona horaria seleccionada (p. ej., Canadá/Este).

8.5.2.3. Configuración del cargador dinámico

De forma predeterminada, el cargador dinámico (`/lib/ld-linux.so.2`) busca en `/usr/lib` las bibliotecas dinámicas que necesitan los programas al ejecutarse. Sin embargo, si existen bibliotecas en directorios distintos a `/usr/lib`, estas deben agregarse al archivo `/etc/ld.so.conf` para que el cargador dinámico las encuentre. Dos directorios que suelen contener bibliotecas adicionales son `/usr/local/lib y /opt/lib`; por lo tanto, agréguelos a la ruta de búsqueda del cargador dinámico.

Cree un nuevo archivo `/etc/ld.so.conf` ejecutando lo siguiente:

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

EOF
```

Si lo desea, el cargador dinámico también puede buscar en un directorio e incluir el contenido de los archivos que se encuentran allí. Generalmente, los archivos en este directorio de inclusión son una línea que especifica la ruta de la biblioteca deseada. Para añadir esta función, ejecute los siguientes comandos:

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

```
# Añadir un directorio de inclusión
include /etc/ld.so.conf.d/*.conf

EOF
mkdir -pv /etc/ld.so.conf.d
```

8.5.3. Contenido de Glibc

Programas instalados:	gencat, getconf, getent, iconv, iconvconfig, ldconfig, ldd, lddlibc4, ld.so (symlink to ld-linux-x86-64.so.2 or ld-linux.so.2), locale, localedef, makedb, mtrace, pcprofiledump, pldd, sln, sotruss, sprof, tzselect, xtrace, zdump, and zic
Bibliotecas instaladas:	ld-linux-x86-64.so.2, ld-linux.so.2, libBrokenLocale.{a,so}, libanl.{a,so}, libc.{a,so}, libc_nonshared.a, libc_malloc_debug.so, libdl.{a,so.2}, libg.a, libm.{a,so}, libmcheck.a, libmemusage.so, libmvec.{a,so}, libnsl.so.1, libnss_compat.so, libnss_dns.so, libnss_files.so, libnss_hesiod.so, libpcprofile.so, libpthread.{a,so.0}, libresolv.{a,so}, librt.{a,so.1}, libthread_db.so, and libutil.{a,so.1}
Directorios instalados:	/usr/include/arpa, /usr/include/bits, /usr/include/gnu, /usr/include/net, /usr/include/netash, /usr/include/netatalk, /usr/include/netax25, /usr/include/neteconet, /usr/include/netinet, /usr/include/netipx, /usr/include/netiucv, /usr/include/netpacket, /usr/include/netrom, /usr/include/netrose, /usr/include/nfs, /usr/include/protocols, /usr/include/rpc, /usr/include/sys, /usr/lib/audit, /usr/lib/gconv, /usr/lib/locale, /usr/libexec/getconf, /usr/share/i18n, /usr/share/zoneinfo, and /var/lib/nss_db

Descripciones breves

Gencat	Genera catálogos de mensajes.
getconf	Muestra los valores de configuración del sistema para las variables específicas del sistema de archivos.
getent	Obtiene entradas de una base de datos administrativa.
iconv	Realiza la conversión del conjunto de caracteres.
iconvconfig	Crea archivos de configuración del módulo **iconv** de carga rápida.
ldconfig	Configura los enlaces de tiempo de ejecución del enlazador dinámico.
ldd	Informa qué bibliotecas compartidas requiere cada programa o biblioteca compartida.
lddlibc4	Asiste a **ldd** con los archivos objeto. No existe en arquitecturas más nuevas como x86_64.
locale	Imprime información diversa sobre la configuración regional actual.

localedef Compila las especificaciones de la configuración regional.

makedb Crea una base de datos simple a partir de una entrada de texto.

mtrace Lee e interpreta un archivo de seguimiento de memoria y muestra un resumen en formato legible.

pcprofiledump Vuelca la información generada por el perfilado de PC.

Pldd Enumera los objetos compartidos dinámicos utilizados por los procesos en ejecución.

sln Un programa **ln** enlazado estáticamente.

sotruss Rastrea las llamadas a procedimientos de bibliotecas compartidas de un comando específico.

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

tzselect Pregunta al usuario sobre la ubicación del sistema e informa la descripción de la zona horaria correspondiente.

xtrace Rastrea la ejecución de un programa imprimiendo la función actualmente ejecutada.

zdump Volcador de zona horaria.

zic Compilador de zona horaria.

`ld-*.so` Programa auxiliar para ejecutables de bibliotecas compartidas.

`libBrokenLocale` Usado internamente por Glibc como un truco para obtener programas dañados (por ejemplo, algunos Motif). aplicaciones) en ejecución. Consulte los comentarios en `glibc-2.41/locale/broken_cur_max.c` para obtener más información.

`libanl` Biblioteca ficticia sin funciones. Anteriormente era la biblioteca de búsqueda de nombres asíncrona, cuyas funciones ahora están en `libc`.

`libc` Biblioteca principal de C.

`libc_malloc_debug` Activa la comprobación de asignación de memoria al precargarse.

`libdl` Biblioteca ficticia sin funciones. Anteriormente era la biblioteca de interfaz de enlace dinámico, cuyas funciones ahora están en `libc`.

`libg` Biblioteca ficticia sin funciones. Anteriormente era una biblioteca de tiempo de ejecución para g++.

`libm` La biblioteca matemática.

`Libmvec` La biblioteca matemática vectorial, enlazada según sea necesario cuando se usa `libm`.

`libmcheck` Activa la comprobación de asignación de memoria cuando se enlaza a.

`libmemusage`	Usada por **memusage** para ayudar a recopilar información sobre el uso de memoria de un programa.
`libnsl`	La biblioteca de servicios de red, ahora obsoleta.
`libnss_*`	Los módulos de conmutación de servicio de nombres, que contienen funciones para resolver nombres de host, nombres de usuario, nombres de grupo, alias, servicios, protocolos, etc. Cargados por `libc` según la configuración en `/etc/nsswitch.conf`.
`libpcprofile`	Se puede precargar en el perfil de PC de un ejecutable.
`libpthread`	Biblioteca ficticia que no contiene funciones. Anteriormente contenía funciones que proporcionaban la mayoría de las interfaces especificadas por las extensiones de subprocesos POSIX.1c y las interfaces de semáforo especificadas por las extensiones de tiempo real POSIX.1b. Ahora, estas funciones se encuentran en `libc`.
`libresolv`	Contiene funciones para crear, enviar e interpretar paquetes a los servidores de nombres de dominio de Internet.
`librt`	Contiene funciones que proporcionan la mayoría de las interfaces especificadas por las extensiones de tiempo real POSIX.1b.
`libthread_db`	Contiene funciones útiles para crear depuradores para programas multiproceso.
`libutil`	Biblioteca ficticia sin funciones. Anteriormente, contenía código para funciones estándar utilizadas en diversas utilidades de Unix. Estas funciones ahora se encuentran en `libc`.

8.6. Zlib-1.3.1

El paquete Zlib contiene rutinas de compresión y descompresión utilizadas por algunos programas.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 6,4 MB

8.6.1. Instalación de Zlib

Preparar Zlib para la compilación:

```
./configure –prefix=/usr
```

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Eliminar una biblioteca estática inútil:

```
rm -fv /usr/lib/libz.a
```

8.6.2. Contenido de Zlib

Bibliotecas instaladas: libz.so

Descripciones breves

`libz` Contiene funciones de compresión y descompresión utilizadas por algunos programas

8.7. Bzip2-1.0.8

El paquete Bzip2 contiene programas para comprimir y descomprimir archivos. Comprimir archivos de texto con **bzip2** ofrece un porcentaje de compresión mucho mejor que con el gzip tradicional.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 7,2 MB

8.7.1. Instalación de Bzip2

Aplique un parche que instale la documentación de este paquete:

```
patch -Np1 -i ../bzip2-1.0.8-install_docs-1.patch
```

El siguiente comando garantiza que la instalación de enlaces simbólicos sea relativa:

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

Asegúrese de que las páginas del manual estén instaladas en la ubicación correcta:

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

Prepare Bzip2 para la compilación con:

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

Significado del parámetro make:

-f Makefile-libbz2_so

Esto hará que Bzip2 se compile utilizando un archivo Makefile diferente, en este caso el archivo Makefile-libbz2_so, que crea una biblioteca dinámica libbz2.so y enlaza las utilidades de Bzip2 con ella. Compilar y probar el paquete:

```
make
```

Instalar los programas:

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

Instalar la biblioteca compartida:

```
cp -av libbz2.so.* /usr/lib
ln -sv libbz2.so.1.0.8 /usr/lib/libbz2.so
```

Instalar el binario **bzip2** compartido en el directorio /usr/bin y reemplazar dos copias de **bzip2** con enlaces simbólicos:

```
cp -v bzip2-shared /usr/bin/bzip2
for i in /usr/bin/{bzcat,bunzip2}; do
  ln -sfv bzip2 $i
done
```

Eliminar una biblioteca estática inútil:

```
rm -fv /usr/lib/libbz2
```

8.7.2. Contenido de Bzip2

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

Bibliotecas instaladas: libbz2.so

Directorio de instalación: /usr/share/doc/bzip2-1.0.8

Descripciones breves

Bunzip2 Descomprime archivos comprimidos

bzcat Descomprime a la salida estándar

bzcmp Ejecuta **cmp** en archivos comprimidos

bzdiff Ejecuta **diff** en archivos comprimidos

bzegrep Ejecuta **egrep** en archivos comprimidos Archivos

bzfgrep Ejecuta **fgrep** en archivos comprimidos

bzgrep Ejecuta **grep** en archivos comprimidos

bzip2 Comprime archivos mediante el algoritmo de compresión de texto Burrows-Wheeler con ordenamiento por bloques y codificación Huffman. La tasa de compresión es mejor que la alcanzada por compresores más convencionales que utilizan algoritmos "Lempel-Ziv", como **gzip**.

Bzip2recover Intenta recuperar datos de archivos comprimidos dañados.

Bzless Se ejecuta con **less** frecuencia en archivos comprimidos

bzmore Se ejecuta con **more** frecuencia en archivos comprimidos

`libbz2` La biblioteca implementa la compresión de datos sin pérdida con ordenamiento por bloques mediante el algoritmo Burrows-Wheeler.

8.8. Xz-5.6.4

El paquete Xz contiene programas para comprimir y descomprimir archivos. Ofrece funciones para los formatos de compresión lzma y xz, que son más recientes. Comprimir archivos de texto con **xz** ofrece un mejor porcentaje de compresión que con los comandos tradicionales **gzip** o **bzip2**.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 21 MB

8.8.1. Instalación de Xz

Prepare Xz para la compilación con:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/xz-5.6.4
```

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.8.2. Contenido de Xz

Programas instalados: lzcat (enlace a xz), lzcmp (enlace a xzdiff), lzdiff (enlace a xzdiff), lzegrep (enlace a xzgrep), lzfgrep (enlace a xzgrep), lzgrep (enlace a xzgrep), lzless (enlace a xzless), lzma (enlace a xz), lzmadec, lzmainfo, lzmore (enlace a xzmore), unlzma (enlace a xz), unxz (enlace a xz), xz, xzcat (enlace a xz), xzcmp (enlace a xzdiff), xzdec, xzdiff, xzegrep (enlace a xzgrep), xzfgrep (enlace a xzgrep), xzgrep, xzless y xzmore

Bibliotecas instaladas: liblzma.so

Directorios instalados: /usr/include/lzma y /usr/share/doc/xz-5.6.4

Descripciones breves

lzcat Descomprime a la salida estándar

lzcmp Ejecuta **cmp** en archivos comprimidos LZMA

lzdiff Ejecuta **diff** en archivos comprimidos LZMA

lzegrep Ejecuta **egrep** en archivos comprimidos LZMA

lzfgrep	Ejecuta **fgrep** en archivos comprimidos LZMA
lzgrep	Ejecuta **grep** en archivos comprimidos LZMA
lzless	Ejecuta **less** en archivos comprimidos LZMA
lzma	Comprime o descomprime archivos con el formato LZMA
lzmadec	Un decodificador pequeño y rápido para archivos comprimidos LZMA
lzmainfo	Muestra la información almacenada en la cabecera del archivo comprimido LZMA
lzmore	Ejecuta **more** en archivos comprimidos LZMA Archivos
unlzma	Descomprime archivos en formato LZMA
unxz	Descomprime archivos en formato XZ
xz	Comprime o descomprime archivos en formato XZ
xzcat	Descomprime a la salida estándar
xzcmp	Ejecuta **cmp** en archivos comprimidos XZ
xzdec	Un decodificador pequeño y rápido para archivos comprimidos XZ
xzdiff	Ejecuta **diff** en archivos comprimidos XZ
xzegrep	Ejecuta **egrep** en archivos comprimidos XZ
xzfgrep	Ejecuta **fgrep** en archivos comprimidos XZ
xzgrep	Ejecuta **grep** en archivos comprimidos XZ
xzless	Se ejecuta **less** en archivos comprimidos XZ
xzmore	Se ejecuta **more** en archivos comprimidos XZ
`liblzma`	La biblioteca que implementa la compresión de datos sin pérdida y con ordenación de bloques, utilizando Algoritmo de cadena Lempel-Ziv-Markov

8.9. Lz4-1.10.0

Lz4 es un algoritmo de compresión sin pérdidas que proporciona una velocidad de compresión superior a 500 MB/s por núcleo. Cuenta con un decodificador extremadamente rápido, con una velocidad de varios GB/s por núcleo. Lz4 es compatible con Zstandard para que ambos algoritmos puedan comprimir los datos más rápidamente.

Tiempo de compilación aproximado: 0,1 SBU
Espacio en discoteca necesario: 4,2MB

8.9.1. Instalación de Lz4

Compilando el paquete:

```
make BUILD_STATIC=no PREFIX=/usr
```

Para probar los resultados, ejecute:

```
hacer -j1 check
```

Instalar el paquete:

```
make BUILD_STATIC=no PREFIX=/usr install
```

8.9.2. Contenido de Lz4

Programas instalados: lz4, lz4c (enlazar a lz4), lz4cat (enlazar a lz4) y unlz4 (enlazar a lz4)

Biblioteca instalada: liblz4.so

Descripciones breves

lz4 Comprimir o descomprimir archivos con el formato LZ4

lz4c Comprimir archivos con el formato LZ4

lz4cat Muestra el contenido de un archivo comprimido con el formato LZ4

unlz4 Descomprime archivos con el formato LZ4

`liblz4` Biblioteca que implementa la compresión de datos sin pérdida mediante el algoritmo LZ4

8.10. Zstd-1.5.7

Zstandard es un algoritmo de compresión en tiempo real que proporciona altas tasas de compresión. Ofrece una amplia gama de ventajas y desventajas entre compresión y velocidad, además de estar respaldado por un decodificador muy rápido.

Tiempo de compilación aproximado: 0.4 SBU
Espacio en disco requerido: 85 MB

8.10.1. Instalación de Zstd

Compilar el paquete:

```
make prefix=/usr
```

Nota

En la salida de la prueba, hay varios lugares que indican "failed". Estos son los esperados y solo "FAIL" es un fallo de prueba real. No debería haber fallos de prueba.

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make prefix=/usr install
```

Eliminar la biblioteca estática:

```
rm -v /usr/lib/libzstd.a
```

8.10.2. Contenido de Zstd

Programas instalados: zstd, zstdcat (enlace a zstd), zstdgrep, zstdless, zstdmt (enlace a zstd) y unzstd (enlace a zstd)

Biblioteca instalada: libzstd.so

Descripciones breves

zstd Comprime o descomprime archivos con el formato ZSTD.

zstdgrep Ejecuta grep en archivos comprimidos con ZSTD.

zstdless Se ejecuta con menos frecuencia en archivos comprimidos con ZSTD.

`libzstd` Biblioteca que implementa la compresión de datos sin pérdida mediante el algoritmo ZSTD.

8.11. Archivo-5.46

El paquete Archivo contiene una utilidad para determinar el tipo de uno o más archivos.

Tiempo de compilación aproximado: Menos de 0.1 SBU
Espacio en disco requerido: 19 MB

8.11.1. Instalación de Archivo

Preparar Archivo para la compilación:

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

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.11.2. Contenido de Archivo

Programas instalados: file

Biblioteca instalada: libmagic.so

Descripciones breves

file Intenta clasificar cada archivo mediante varias pruebas: pruebas del sistema de archivos, pruebas de números mágicos y pruebas de lenguaje.

`libmagic` Contiene rutinas para el reconocimiento de números mágicos, utilizadas por el programa Archivo.

8.12. Readline-8.2.13

El paquete Readline es un conjunto de bibliotecas que ofrecen funciones de edición e historial desde la línea de comandos.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 16 MB

8.12.1. Instalación de Readline

Reinstalar Readline provocará que las bibliotecas antiguas se trasladen a <libraryname>.old. Aunque esto no suele ser un problema, en algunos casos puede provocar un error de enlace en **ldconfig**. Esto se puede evitar ejecutando los dos siguientes comandos sed:

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

Evitar la codificación rígida de las rutas de búsqueda de bibliotecas (rpath) en las bibliotecas compartidas. Este paquete no necesita rpath para instalarse en la ubicación estándar, y rpath puede causar efectos no deseados o incluso problemas de seguridad:

```
sed -i 's/-Wl,-rpath,[^ ]*//' support/shobj-conf
```

Preparar Readline para la compilación:

```
./configure --prefix=/usr      \
            --disable-static   \
            --with-curses      \
            --docdir=/usr/share/doc/readline-8.2.13
```

Significado de la nueva opción de configuración:

`--with-curses`

Esta opción indica a Readline que puede encontrar las funciones de la biblioteca termcap en la biblioteca curses, no en una biblioteca termcap independiente. Esto generará el archivo readline.pc correcto.

Compilar el paquete:

```
make SHLIB_LIBS="-lncursesw"
```

Significado de la opción "make":

`SHLIB_LIBS="-lncursesw"`

Esta opción obliga a Readline a enlazar con la biblioteca `libncursesw`. Para más información, consulte la sección "Bibliotecas compartidas" en el archivo `README` del paquete.

Este paquete no incluye un conjunto de pruebas.

Instalar el paquete:

```
make install
```

Si lo desea, instale la documentación:

```
install -v -m644 doc/*.{ps,pdf,html,dvi} /usr/share/doc/readline-8.2.13
```

8.12.2. Contenido de Readline

Bibliotecas instaladas: libhistory.so y libreadline.so
Directorios instalados: /usr/include/readline y /usr/share/doc/readline-8.2.13

Descripciones breves

`libhistory` Proporciona una interfaz de usuario consistente para recuperar líneas del historial.

`libreadline` Proporciona un conjunto de comandos para manipular el texto introducido en una sesión interactiva de un programa.

8.13. M4-1.4.19

El paquete M4 contiene un procesador de macros.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 49 MB

8.13.1. Instalación de M4

Preparar M4 para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.13.2. Contenido de M4

Programa instalado: m4

Descripciones breves

m4 Copia los archivos indicados y expande las macros que contienen. Estas macros pueden estar integradas o definidas por el usuario y pueden aceptar cualquier número de argumentos. Además de la expansión de macros, **m4** cuenta con funciones integradas para incluir archivos con nombre, ejecutar comandos Unix, realizar operaciones aritméticas con enteros, manipular texto y realizar recursiones, entre otras. El programa **m4** puede utilizarse como interfaz para un compilador o como procesador de macros por sí mismo.

8.14. Bc-7.0.3

El paquete Bc contiene un lenguaje de procesamiento numérico de precisión arbitraria.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 7,8 MB

8.14.1. Instalación de Bc

Preparar Bc para la compilación:

```
CC=gcc ./configure --prefix=/usr -G -O3 -r
```

Significado de las opciones de configuración:

CC=gcc

Este parámetro especifica el compilador a utilizar.

-G

Omite las partes del conjunto de pruebas que no funcionarán hasta que se instale el programa bc.

-O3

Especifica la optimización a utilizar.

-r

Habilita el uso de Readline para mejorar la función de edición de líneas de bc.

Compila el paquete:

```
make
```

Para probar bc, ejecuta:

```
make test
```

Instala el paquete:

```
make install
```

8.14.2. Contenido de Bc

Programas instalados: bc y dc

Descripciones breves

bc Calculadora de línea de comandos

dc Calculadora de línea de comandos con pulido inverso

8.15. Flex-2.6.4

El paquete Flex contiene una utilidad para generar programas que reconocen patrones en texto.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 33 MB

8.15.1. Instalación de Flex

Preparar Flex para la compilación:

```
./configure --prefix=/usr \
            --docdir=/usr/share/doc/flex-2.6.4 \
            --disable-static
```

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Algunos programas aún no conocen **flex** e intentan ejecutar su predecesor, **lex**. Para dar soporte a estos programas, cree un enlace simbólico llamado lex que ejecute **flex** en modo de emulación de **lex** y cree también la página de manual de **lex** como enlace simbólico:

```
ln -sv flex   /usr/bin/lex
ln -sv flex.1 /usr/share/man/man1/lex.1
```

8.15.2. Contenido de Flex

Programas instalados: `flex, flex++ (enlace a flex) y lex (enlace a flex)`
Bibliotecas instaladas: `libfl.so`
Directorio de instalación: `/usr/share/doc/flex-2.6.4`

Descripciones breves

flex Herramienta para generar programas que reconocen patrones en texto. Permite la versatilidad de especificar las reglas para la búsqueda de patrones, eliminando la necesidad de desarrollar un programa especializado.

flex++ Extensión de flex, utilizada para generar código y clases en C++. Es un enlace simbólico a **flex**

lex Un enlace simbólico que ejecuta **flex** en modo de emulación **lex**

libfl La biblioteca `flex`

8.16. Tcl-8.6.16

El paquete Tcl contiene el Lenguaje de Comandos de Herramientas, un lenguaje de scripting robusto y de propósito general. El paquete Expect está escrito en Tcl (se pronuncia "cosquillas").

Tiempo de compilación aproximado: 3.1 SBU
Espacio en disco requerido: 91 MB

8.16.1. Instalación de Tcl

Este paquete y los dos siguientes (Expect y DejaGNU) se instalan para permitir la ejecución de las suites de pruebas de Binutils, GCC y otros paquetes. Instalar tres paquetes para realizar pruebas puede parecer excesivo, pero es muy tranquilizador, si no esencial, saber que las herramientas más importantes funcionan correctamente.

Preparar Tcl para la compilación:

```
SRCDIR=$(pwd)
cd unix
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --disable-rpath
```

Significado de los nuevos parámetros de configuración:

`--disable-rpath`

Este parámetro evita la codificación rígida de las rutas de búsqueda de bibliotecas (rpath) en los archivos binarios ejecutables y las bibliotecas compartidas. Este paquete no necesita rpath para la instalación en la ubicación estándar, y rpath puede, en ocasiones, causar efectos no deseados o incluso problemas de seguridad.

Construir el paquete:

```
Make

sed -e "s|$SRCDIR/unix|/usr/lib|" \
    -e "s|$SRCDIR|/usr/include|"  \
    -i tclConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/tdbc1.1.10|/usr/lib/tdbc1.1.10|"  \
    -e "s|$SRCDIR/pkgs/tdbc1.1.10/generic|/usr/include|"     \
    -e "s|$SRCDIR/pkgs/tdbc1.1.10/library|/usr/lib/tcl8.6|"  \
    -e "s|$SRCDIR/pkgs/tdbc1.1.10|/usr/include|"             \
    -i pkgs/tdbc1.1.10/tdbcConfig.sh

sed -e "s|$SRCDIR/unix/pkgs/itcl4.3.2|/usr/lib/itcl4.3.2|"   \
    -e "s|$SRCDIR/pkgs/itcl4.3.2/generic|/usr/include|"      \
    -e "s|$SRCDIR/pkgs/itcl4.3.2|/usr/include|"              \
    -i pkgs/itcl4.3.2/itclConfig.sh

unset SRCDIR
```

Las distintas instrucciones "sed" después del comando "make" eliminan las referencias al directorio de compilación de los archivos de configuración y las reemplazan con el directorio de instalación. Esto no es obligatorio para el resto de LFS, pero puede ser necesario si un paquete compilado posteriormente utiliza Tcl.

Para probar los resultados, ejecute:

```
make test
```

Instalar el paquete:

```
make install
```

Permite la escritura en la biblioteca instalada para que los símbolos de depuración se puedan eliminar posteriormente:

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

Instalar las cabeceras de Tcl. El siguiente paquete, Expect, las requiere.

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

Ahora cree el enlace simbólico necesario:

```
ln -sfv tclsh8.6 /usr/bin/tclsh
```

Renombrar una página de manual que entre en conflicto con una página de manual de Perl:

```
mv /usr/share/man/man3/{Thread,Tcl_Thread}.3
```

Opcionalmente, instale la documentación con los siguientes comandos:

```
cd ..
tar -xf ../tcl8.6.16-html.tar.gz --strip-components=1
mkdir -v -p /usr/share/doc/tcl-8.6.16
cp -v -r ./html/* /usr/share/doc/tcl-8.6.16
```

8.16.2. Contenido de Tcl

Programas instalados: tclsh (enlace a tclsh8.6) y tclsh8.6
Bibliotecas instaladas: libtcl8.6.so y libtclstub8.6.a

Descripciones breves

tclsh8.6	La consola de comandos de Tcl
tclsh	Un enlace a tclsh8.6
libtcl8.6.so	La biblioteca de Tcl
libtclstub8.6.a	La biblioteca de stubs de Tcl

8.17. Expect-5.45.4

El paquete Expect contiene herramientas para automatizar, mediante diálogos con scripts, aplicaciones interactivas como Telnet, FTP, Passwd, FSC, Rlogin y Tip. Expect también es útil para probar estas mismas aplicaciones, así como para facilitar todo tipo de tareas que resultan extremadamente difíciles con cualquier otro sistema. El framework DejaGnu está escrito en Expect.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 3.9 MB

8.17.1. Instalación de Expect

Expect necesita PTY para funcionar. Verifique que los PTY funcionen correctamente dentro del entorno chroot realizando una prueba sencilla:

```
python3 -c 'from pty import spawn; spawn(["echo", "ok"])'
```

Este comando debería tener un resultado correcto. Si, por el contrario, el resultado incluye `OSError: out of pty devices`, el entorno no está configurado para el correcto funcionamiento de PTY. Debe salir del entorno chroot, leer de nuevo la Sección 7.3, "Preparación de los sistemas de archivos virtuales del kernel" y asegurarse de que el sistema de archivos `devpts` (y otros sistemas de archivos virtuales del kernel) estén montados correctamente. A continuación, vuelva a entrar en el entorno chroot siguiendo la Sección 7.4, "Acceso al entorno chroot". Este problema debe resolverse antes de continuar; de lo contrario, las suites de pruebas que requieren Expect (por ejemplo, las suites de pruebas de Bash, Binutils, GCC, GDBM y, por supuesto, el propio Expect) fallarán gravemente y podrían producirse otras fallas menores. Ahora, realice algunos cambios para permitir el paquete con gcc-14.1 o posterior:

```
patch -Np1 -i ../expect-5.45.4-gcc14-1.patch
```

Preparar Expect para la compilación:

```
./configure --prefix=/usr           \
            --with-tcl=/usr/lib     \
            --enable-shared         \
            --disable-rpath         \
            --mandir=/usr/share/man \
            --with-tclinclude=/usr/include
```

Significado de las opciones de configuración:

`--with-tcl=/usr/lib`

Este parámetro es necesario para indicar a Configure dónde se encuentra el script tclConfig.sh.

`--with-tclinclude=/usr/include`

Esto indica explícitamente a Expect dónde encontrar las cabeceras internas de Tcl.

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make test
```

Instalar el paquete:

```
make install
ln -svf expect5.45.4/libexpect5.45.4.so /usr/lib
```

8.17.2. Contenido de Expect

Programa instalado: expect
Biblioteca instalada: libexpect5.45.4.so

Descripciones breves

expect Se comunica con otros programas interactivos mediante un script

`libexpect-5.45.4.so` Contiene funciones que permiten usar Expect como extensión de Tcl o directamente desde C o C++ (sin Tcl)

8.18. DejaGNU-1.6.3

El paquete DejaGnu contiene un marco de trabajo para ejecutar conjuntos de pruebas en herramientas GNU. Está escrito en expect, que a su vez utiliza Tcl (lenguaje de comandos de herramientas).

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 6,9 MB

8.18.1. Instalación de DejaGNU

El desarrollador original recomienda compilar DejaGNU en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Preparar DejaGNU para la compilación:

```
../configure --prefix=/usr
makeinfo ––html --no-split -o doc/dejagnu.html ../doc/dejagnu.texi
makeinfo ––plaintext       -o doc/dejagnu.txt  ../doc/dejagnu.texi
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
install -v -dm755 /usr/share/doc/dejagnu-1.6.3
install -v -m644  doc/dejagnu.{html,txt} /usr/share/doc/dejagnu-1.6.3
```

8.18.2. Contenido de DejaGNU

Programas instalados: dejagnu y runtest

Descripciones breves

dejagnu Lanzador de comandos auxiliar de DejaGNU

runtest Un script contenedor que localiza la shell de expect adecuada y luego ejecuta DejaGNU

8.19. Pkgconf-2.3.0

El paquete pkgconf es el sucesor de pkg-config y contiene una herramienta para pasar la ruta de inclusión y/o las rutas de biblioteca a las herramientas de compilación durante las fases de configuración y creación de la instalación de paquetes.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 4,7 MB

8.19.1. Instalación de Pkgconf

Preparar Pkgconf para la compilación:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/pkgconf-2.3.0
```

Compilar el paquete:

```
make
```

Instalar el paquete:

```
make install
```

Para mantener la compatibilidad con el Pkg-config original, cree dos enlaces simbólicos:

```
ln -sv pkgconf    /usr/bin/pkg-config
ln -sv pkgconf.1 /usr/share/man/man1/pkg-config.1
```

8.19.2. Contenido de Pkgconf

Programas instalados pkgconf, pkg-config (enlace a pkgconf) y bomtool
Biblioteca instalada: libpkgconf.so
Directorio de instalación: /usr/share/doc/pkgconf-2.3.0

Descripciones breves

pkgconf Devuelve metainformación para la biblioteca o paquete especificado.

bomtool Genera una lista de materiales de software a partir de los archivos .pc de pkg-config.

`libpkgconf` Contiene la mayor parte de la funcionalidad de pkgconf y permite que otras herramientas, como IDE y compiladores, utilicen sus frameworks.

8.20. Binutils-2.44

El paquete Binutils contiene un enlazador, un ensamblador y otras herramientas para gestionar archivos objeto.

Tiempo de compilación aproximado: 1.6 SBU
Espacio en disco requerido: 819 MB

8.20.1. Instalación de Binutils

La documentación de Binutils recomienda compilar Binutils en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Preparar Binutils para la compilación:

```
../configure --prefix=/usr       \
             --sysconfdir=/etc   \
             --enable-ld=default \
             --enable-plugins    \
             --enable-shared     \
             --disable-werror    \
             --enable-64-bit-bfd \
             --enable-new-dtags  \
             --with-system-zlib  \
             --enable-default-hash-style=gnu
```

Significado de los nuevos parámetros de configuración:

`--enable-ld=default`

Compile el enlazador bfd original e instálelo como ld (el enlazador predeterminado) y ld.bfd.

`--enable-plugins`

Habilita la compatibilidad con plugins para el enlazador.

`--with-system-zlib`

Use la biblioteca zlib instalada en lugar de compilar la versión incluida.

Compilar el paquete:

```
make tooldir=/usr
```

Significado del parámetro make:

`tooldir=/usr`

Normalmente, el directorio de herramientas (el directorio donde se ubicarán los ejecutables) se establece en $(exec_prefix)/$(target_alias). Por ejemplo, las máquinas x86_64 lo expandirían a /usr/x86_64-pc-linux-gnu. Al tratarse de un sistema personalizado, este directorio específico de destino en /usr no es necesario. $(exec_prefix)/$(target_alias) se usaría si el sistema se usara

para realizar compilación cruzada (por ejemplo, compilar un paquete en una máquina Intel que genera código ejecutable en máquinas PowerPC).

Importante

`El conjunto de pruebas para Binutils de esta sección se considera crítico. No lo omita bajo ninguna circunstancia.`

Pruebe los resultados:

```
make -k check
```

Para obtener una lista de las pruebas fallidas, ejecute:

```
grep '^FAIL:' $(find -name '*.log')
```

Instale el paquete:

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

Elimine las bibliotecas estáticas innecesarias y otros archivos:

```
rm -rfv /usr/lib/lib{bfd,ctf,ctf-nobfd,gprofng,opcodes,sframe}.a \
        /usr/share/doc/gprofng/
```

8.20.2. Contenido de Binutils

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

Bibliotecas instaladas: libbfd.so, libctf.so, libctf-nobfd.so, libgprofng.so, libopcodes.so y libsframe.so

Directorio de instalación: /usr/lib/ldscripts

Descripciones breves

addr2line Traduce las direcciones de los programas a nombres de archivo y números de línea; Dada una dirección y el nombre de un ejecutable, utiliza la información de depuración del ejecutable para determinar qué archivo fuente y número de línea están asociados a la dirección.

ar Crea, modifica y extrae de archivos.

as Un ensamblador que ensambla la salida de gcc en archivos objeto.

c++filt Utilizado por el enlazador para descomponer símbolos de C++ y Java y evitar que las funciones sobrecargadas entren en conflicto.

dwp La utilidad de empaquetado DWARF.

elfedit Actualiza las cabeceras ELF de los archivos ELF.

gprof	Muestra datos del perfil del gráfico de llamadas.
gprofng	Recopila y analiza datos de rendimiento.
ld	Un enlazador que combina varios archivos objeto y de archivo en un solo archivo, reubicando sus datos y enlazando referencias de símbolos.
ld.bfd	Un enlace duro a **ld**.
nm	Enumera los símbolos que aparecen en un archivo objeto dado.
objcopy	Traduce un tipo de archivo objeto a otro.
objdump	Muestra información sobre el archivo objeto dado, con opciones que controlan la información específica que se mostrará; La información mostrada es útil para los programadores que trabajan en las herramientas de compilación.
ranlib	Genera un índice del contenido de un archivo y lo almacena en él; el índice enumera todos los símbolos definidos por los miembros del archivo que son archivos objeto reubicables.
readelf	Muestra información sobre los binarios de tipo ELF.
size	Enumera los tamaños de sección y el tamaño total de los archivos objeto dados.
strings	Genera, para cada archivo, las secuencias de caracteres imprimibles con al menos la longitud especificada (cuatro por defecto); para archivos objeto, imprime, por defecto, solo las cadenas de las secciones de inicialización y carga, mientras que para otros tipos de archivos, escanea el archivo completo.
strip	Descarta símbolos de los archivos objeto.
`libbfd`	La biblioteca de descriptores de archivos binarios.
`libctf`	La biblioteca de soporte de depuración de formatos de tipo ANSI-C compatibles.
`libctf-nobfd`	Una variante de libctf que no utiliza la funcionalidad de libbfd.
`libgprofng`	Una biblioteca que contiene la mayoría de las rutinas utilizadas por **gprofng**.
`libopcodes`	Una biblioteca para gestionar códigos de operación (las versiones de "texto legible" de las instrucciones para el procesador). Se utiliza para crear utilidades como **objdump**.
`libsframe`	Una biblioteca para soportar el rastreo inverso en línea utilizando un desempaquetador (unwinder) simple..

8.21. GMP-6.3.0

El paquete GMP contiene bibliotecas matemáticas. Estas ofrecen funciones útiles para cálculos aritméticos de precisión arbitraria.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 54 MB

8.21.1. Instalación de GMP

Nota

Si está compilando para x86 de 32 bits, pero tiene una CPU capaz de ejecutar código de 64 bits y ha especificado CFLAGS en el entorno, el script de configuración intentará configurarlo para 64 bits y fallará. Evite esto invocando el comando de configuración a continuación con

```
ABI=32 ./configure ...
```

Nota

La configuración predeterminada de GMP genera bibliotecas optimizadas para el procesador host. Si se desean bibliotecas adecuadas para procesadores con menor capacidad que la CPU del host, se pueden crear bibliotecas genéricas añadiendo la opción --host=none-linux-gnu al comando de **configuración**.

Preparar GMP para la compilación:

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.3.0
```

Significado de las nuevas opciones de configuración:

--enable-cxx

Este parámetro habilita la compatibilidad con C++.

--docdir=/usr/share/doc/gmp-6.3.0

Esta variable especifica la ubicación correcta de la documentación.

Compilar el paquete y generar la documentación HTML:

```
make
make html
```

Importante

El conjunto de pruebas para GMP de esta sección se considera crítico. No lo omita bajo ninguna circunstancia.

Pruebe los resultados:

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

Precaución

El código de gmp está altamente optimizado para el procesador donde se compila. Ocasionalmente, el código que detecta el procesador identifica erróneamente las capacidades del sistema, lo que genera errores e las pruebas u otras aplicaciones que utilizan las bibliotecas gmp con el mensaje "*Illegal instruction*. En este caso, se debe reconfigurar gmp con la opción *--host=none-linux-gnu* y reconstruir.

Asegúrese de que al menos 199 pruebas del conjunto de pruebas hayan pasado. Compruebe los resultados con el siguiente comando:

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

Instale el paquete y su documentación:

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

8.21.2. Contenido de GMP

Bibliotecas instaladas: libgmp.so y libgmpxx.so

Directorio de instalación: /usr/share/doc/gmp-6.3.0

Descripciones breves

`libgmp` Contiene funciones matemáticas de precisión

`libgmpxx` Contiene funciones matemáticas de precisión de C++

8.22. MPFR-4.2.1

El paquete MPFR contiene funciones para cálculos matemáticos de precisión múltiple.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 43 MB

8.22.1. Instalación de MPFR

Preparar MPFR para la compilación:

```
./configure --prefix=/usr        \
            --disable-static     \
            --enable-thread-safe \
            --docdir=/usr/share/doc/mpfr-4.2.1
```

Compilar el paquete y generar la documentación HTML:

```
make
make html
```

Importante

El conjunto de pruebas para MPFR de esta sección se considera crítico. No lo omita bajo ninguna circunstancia.

Pruebe los resultados y asegúrese de que las 198 pruebas hayan sido aprobadas:

```
make check
```

Instalar el paquete y su documentación:

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

8.22.2. Contenido de MPFR

Bibliotecas instaladas: libmpfr.so
Directorio de instalación: /usr/share/doc/mpfr-4.2.1

Descripciones breves

`libmpfr` Contiene funciones matemáticas de precisión múltiple

8.23. MPC-1.3.1

El paquete MPC contiene una biblioteca para la aritmética de números complejos con precisión arbitrariamente alta y redondeo correcto del resultado.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 22 MB

8.23.1. Instalación de MPC

Preparar MPC para la compilación:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/mpc-1.3.1
```

Compilar el paquete y generar la documentación HTML:

```
make
make html
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete y su documentación:

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

8.23.2. Contenido de MPC

Bibliotecas instaladas: libmpc.so
Directorio de instalación: /usr/share/doc/mpc-1.3.1

Descripciones breves

libmpc	Contiene funciones matemáticas complejas

8.24. Attr-2.5.2

El paquete Attr contiene utilidades para administrar los atributos extendidos de los objetos del sistema de archivos.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 4,1 MB

8.24.1. Instalación de Attr

Preparar Attr para la compilación:

```
./configure --prefix=/usr     \
            --disable-static  \
            --sysconfdir=/etc \
            --docdir=/usr/share/doc/attr-2.5.2
```

Compilar el paquete:

```
make
```

Las pruebas deben ejecutarse en un sistema de archivos compatible con atributos extendidos, como ext2, ext3 o ext4. Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.24.2. Contenido de Attr

Programas instalados: attr, getfattr y setfattr
Biblioteca instalada: libattr.so
Directorios instalados: /usr/include/attr y /usr/share/doc/attr-2.5.2

Descripciones breves

attr Extiende los atributos de los objetos del sistema de archivos.

getfattr Obtiene los atributos extendidos de los objetos del sistema de archivos.

setfattr Establece los atributos extendidos de los objetos del sistema de archivos.

`libattr` Contiene las funciones de la biblioteca para manipular atributos extendidos.

8.25. Acl-2.3.2

El paquete Acl contiene utilidades para administrar las Listas de Control de Acceso (ACL), que se usan para definir permisos discrecionales detallados de acceso a archivos y directorios.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 6,5 MB

8.25.1. Instalación de Acl

Preparar Acl para la compilación:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/acl-2.3.2
```

Compilar el paquete:

```
make
```

Las pruebas de Acl deben ejecutarse en un sistema de archivos compatible con controles de acceso. Para comprobar los resultados, ejecute:

```
make check
```

Se sabe que una prueba llamada `test/cp.test` falla porque Coreutils aún no está compilado con la compatibilidad de Acl.

Instalar el paquete:

```
make install
```

8.25.2. Contenido de Acl

Programas instalados: `chacl, getfacl y setfacl`
Biblioteca instalada: `libacl.so`
Directorios instalados: `/usr/include/acl y /usr/share/doc/acl-2.3.2`

Descripciones breves

chacl Cambia la lista de control de acceso de un archivo o directorio

getfacl Obtiene listas de control de acceso a archivos

setfacl Establece listas de control de acceso a archivos

`libacl` Contiene las funciones de la biblioteca para manipular las listas de control de acceso

8.26. Libcap-2.73

El paquete Libcap implementa la interfaz de espacio de usuario con las capacidades POSIX 1003.1e disponibles en los kernels de Linux.

Estas capacidades dividen el privilegio root en un conjunto de privilegios distintos.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 3,0 MB

8.26.1. Instalación de Libcap

Evitar la instalación de bibliotecas estáticas:

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

Compilar el paquete:

```
make prefix=/usr lib=lib
```

Significado de la opción make:

lib=lib

Este parámetro establece el directorio de la biblioteca en /usr/lib en lugar de /usr/lib64 en x86_64. No tiene efecto en x86. Para probar los resultados, ejecute:

```
make test
```

Instale el paquete:

```
make prefix=/usr lib=lib install
```

8.26.2. Contenido de Libcap

Programas instalados: capsh, getcap, getpcaps y setcap
Biblioteca instalada: libcap.so y libpsx.so

Descripciones breves

capsh Un contenedor de shell para explorar y restringir la compatibilidad con capacidades

getcap Examina las capacidades del archivo

getpcaps Muestra las capacidades del proceso o procesos consultados

setcap Establece las capacidades del archivo

`libcap` Contiene las funciones de la biblioteca para manipular las capacidades de POSIX 1003.1e

`libpsx` Contiene funciones para la compatibilidad con la semántica POSIX para las llamadas al sistema asociadas con la biblioteca pthread

8.27. Libxcrypt-4.4.38

El paquete Libxcrypt contiene una biblioteca moderna para el hash unidireccional de contraseñas.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 12 MB

8.27.1. Instalación de Libxcrypt

Preparar Libxcrypt para la compilación:

```
./configure --prefix=/usr                \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=no     \
            --disable-static             \
            --disable-failure-tokens
```

Significado de las nuevas opciones de configuración:

`--enable-hashes=strong,glibc`

Construir algoritmos hash robustos recomendados para casos de seguridad y los algoritmos hash proporcionados por libcrypt Glibc tradicional para compatibilidad.

`--enable-obsolete-api=no`

Desactivar funciones obsoletas de la API. No son necesarios para un sistema Linux moderno compilado desde el código fuente.

`--disable-failure-tokens`

Desactiva la función de token de error. Es necesaria para la compatibilidad con las bibliotecas hash tradicionales de algunas plataformas, pero un sistema Linux basado en Glibc no la necesita.

Compila el paquete:

```
make
```

Para probar los resultados, ejecuta:

```
make check
```

Instala el paquete:

```
make install
```

Nota

Las instrucciones anteriores deshabilitaron funciones obsoletas de la API, ya que ningún paquete instalado mediante la compilación desde el código fuente las

```
enlazaría en tiempo de ejecución. Sin embargo, las únicas aplicaciones binarias
conocidas que enlazan con estas funciones requieren la versión 1 de ABI. Si
necesita estas funciones debido a alguna aplicación binaria o para cumplir con
LSB, vuelva a compilar el paquete con los siguientes comandos:
```

```
make distclean
./configure --prefix=/usr                \
            --enable-hashes=strong,glibc \
            --enable-obsolete-api=glibc  \
            --disable-static             \
            --disable-failure-tokens
make
cp -av --remove-destination .libs/libcrypt.so.1* /usr/lib
```

8.27.2. Contenido de Libxcrypt

Bibliotecas instaladas: libcrypt.so

Descripciones breves

`libcrypt` Contiene funciones para generar hashes de contraseñas

8.28. Shadow-4.17.3

El paquete Shadow contiene programas para gestionar contraseñas de forma segura.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 114 MB

8.28.1. Instalación de Shadow

Importante

Si ha instalado Linux-PAM, debe seguir las instrucciones de BLFS en lugar de esta página para compilar (o reconstruir o actualizar) Shadow.

Nota

Si desea forzar el uso de contraseñas seguras, primero instale y configure Linux-PAM. A continuación, instale y configure Shadow con compatibilidad con PAM. Por último, instale libpwquality y configure PAM para usarlo.

Desactive la instalación del programa **groups** y sus páginas de manual, ya que Coreutils ofrece una versión mejorada. Además, evite la instalación de páginas de manual que ya estaban instaladas en la Sección 8.3, “Man-pages-6.12”:

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

En lugar de usar el método de cifrado predeterminado, utiliza el método de cifrado de contraseñas *YESCRYPT*, mucho más seguro, que además permite contraseñas de más de 8 caracteres. También es necesario cambiar la ubicación obsoleta `/var/spool/mail` (que Shadow usa por defecto para los buzones de correo de usuario) por la ubicación `/var/mail`, que es la utilizada actualmente. Por último, elimina `/bin` y `/sbin` del `PATH`, ya que son simplemente enlaces simbólicos a sus respectivos equivalentes en `/usr`.

Advertencia

Incluir /bin y/o /sbin en la variable de entorno PATH puede provocar que algunos paquetes de BLFS fallen al compilar, así que no lo hagas ni en el archivo .bashrc ni en ningún otro lugar.

```
sed -e 's:#ENCRYPT_METHOD DES:ENCRYPT_METHOD YESCRYPT:' \
    -e 's:/var/spool/mail:/var/mail:'                   \
    -e '/PATH=/{s@/sbin:@@;s@/bin:@@}'                  \
    -i etc/login.defs
```

Preparar Shadow para la compilación:

```
touch /usr/bin/passwd
./configure --sysconfdir=/etc\
            --disable-static\
            --with-{b,yes}crypt \
            --without-libbsd\
            --with-group-name-max-length=32
```

Significado de las nuevas opciones de configuración:

touch /usr/bin/passwd

El archivo `/usr/bin/passwd` debe existir porque su ubicación está codificada en algunos programas; si no existe, el script de instalación lo creará en el lugar incorrecto.

`--with-{b,yes}crypt`

El shell amplía esto a dos opciones: `--with-bcrypt` y `--with-yescrypt`. Estas permiten que shadow utilice los algoritmos Bcrypt y Yescrypt implementados por Libxcrypt para el hash de contraseñas. Estos algoritmos son más seguros (en particular, mucho más resistentes a ataques basados en GPU) que los algoritmos SHA tradicionales.

`--with-group-name-max-length=32`

El nombre de usuario máximo permitido es de 32 caracteres. Mantenga la longitud máxima del nombre de grupo igual.

`--without-libbsd`

No utilice la función readpassphrase de libbsd, que no está en LFS. Utilice la copia interna en su lugar.

Compilar el paquete:

```
make
```

Este paquete no incluye un conjunto de pruebas.

Instalar el paquete:

```
make exec_prefix=/usr install
make -C man install-man
```

8.28.2. Configuración de Shadow

Este paquete contiene utilidades para agregar, modificar y eliminar usuarios y grupos. establecer y cambiar sus contraseñas; y realizar otras tareas administrativas. Para obtener una explicación completa del significado del shadowing de contraseñas, consulte el archivo `doc/HOWTO` dentro del árbol de fuentes descomprimido. Si utiliza compatibilidad con Shadow, tenga en cuenta que los programas que necesitan verificar contraseñas (administradores de pantalla, programas FTP, daemons pop3, etc.) deben ser compatibles con Shadow. Es decir, deben poder trabajar con contraseñas shadowed.

Para habilitar las contraseñas shadowed, ejecute el siguiente comando:

```
pwconv
```

Para habilitar las contraseñas de grupo shadowed, ejecute:

```
grpconv
```

La configuración predeterminada de Shadow para la utilidad **useradd** requiere una explicación. Primero, la acción predeterminada de la utilidad **useradd** es crear el usuario y un grupo con el mismo nombre que el usuario. De forma predeterminada, los números de ID de usuario (UID) e ID de grupo (GID) comienzan en 1000. Esto significa que, si no se pasan parámetros adicionales a **useradd**, cada usuario será miembro de un grupo único en el sistema. Si este comportamiento no es el deseado, deberá pasar el parámetro -g o -N a **useradd**, o bien cambiar la configuración de *USERGROUPS_ENAB* en `/etc/login.defs`. Consulte *useradd(8)* para obtener más información.

En segundo lugar, para cambiar los parámetros predeterminados, debe crear el archivo `/etc/default/useradd` y adaptarlo a sus necesidades específicas. Créelo con:

```
mkdir -p /etc/default
useradd -D --gid 999
```

Explicación de los parámetros de /etc/default/useradd

`GROUP=999`

Este parámetro establece el inicio de los números de grupo utilizados en el archivo /`etc/group`. El valor 999 proviene del parámetro `--gid` anterior. Puede configurarlo con cualquier valor. Tenga en cuenta que **useradd** nunca reutilizará un UID o GID. Si se utiliza el número identificado en este parámetro, se utilizará el siguiente número disponible. Tenga en cuenta también que si no tiene un grupo con un ID igual a este número en su sistema, la primera vez que use **useradd** sin el parámetro -g, se generará un mensaje de error: **useradd**: GID desconocido 999, aunque la cuenta se haya creado correctamente. Por eso creamos el grupo "users" con este ID en la Sección 7.6, "Creación de archivos esenciales y enlaces simbólicos".

`CREATE_MAIL_SPOOL=yes`

Este parámetro hace que useradd cree un archivo de buzón para cada nuevo usuario. useradd asignará la propiedad de este archivo al grupo de `correo` con permisos 0660. Si prefiere no crear estos archivos, ejecute el siguiente comando:

```
sed -i '/MAIL/s/yes/no/' /etc/default/useradd
```

8.28.3. Configuración de la contraseña de root

Elija una contraseña para el usuario root y configúrela ejecutando:

```
passwd root
```

8.28.4. Contenido de Shadow

Programas instalados: chage, chfn, chgpasswd, chpasswd, chsh, expiry, faillog, getsubids, gpasswd, groupadd, groupdel, groupmems, groupmod, grpck, grpconv, grpunconv, login, logoutd, newgidmap, newgrp, newuidmap, newusers, nologin, passwd, pwck, pwconv, pwunconv, sg (link to newgrp), su, useradd, userdel, usermod, vigr (link to vipw), and vipw

Directorios instalados: /etc/default and /usr/include/shadow

Bibliotecas instaladas: libsubid.so

Descripciones breves

chage Se utiliza para cambiar el número máximo de días entre cambios obligatorios de contraseña.

chfn Se utiliza para cambiar la contraseña completa de un usuario. Nombre y otra información

chgpasswd Se utiliza para actualizar las contraseñas de grupo por lotes.

chpasswd Se utiliza para actualizar las contraseñas de usuario por lotes.

chsh Se utiliza para cambiar el shell de inicio de sesión predeterminado de un usuario.

expiry Comprueba y aplica la política actual de expiración de contraseñas.

faillog Se utiliza para examinar el registro de errores de inicio de sesión, establecer un número máximo de errores antes de que una cuenta se bloquee y restablecer el recuento de errores.

getsubids Se utiliza para listar los rangos de ID subordinados de un usuario.

gpasswd Se utiliza para agregar y eliminar miembros y administradores a grupos.

groupadd Crea un grupo con el nombre especificado.

groupdel Elimina el grupo con el nombre especificado.

groupmems Permite a un usuario administrar su propia lista de miembros del grupo sin necesidad de privilegios de superusuario.

groupmod Se utiliza para modificar el nombre o GID del grupo dado.

grpck Verifica la integridad de los archivos de grupo `/etc/group` y `/etc/gshadow`.

grpconv Crea o actualiza el archivo de grupo shadow a partir del archivo de grupo normal.

grpunconv Actualiza `/etc/group` desde `/etc/gshadow` y luego elimina este último.

login El sistema lo utiliza para permitir que los usuarios inicien sesión.

logoutd	Es un demonio que se utiliza para imponer restricciones en el tiempo y los puertos de inicio de sesión.
newgidmap	Se usa para establecer el mapeo de GID de un espacio de nombres de usuario
newgrp	Se usa para cambiar el GID actual durante una sesión de inicio de sesión
newuidmap	Se usa para establecer el mapeo de UID de un espacio de nombres de usuario
newusers	Se usa para crear o actualizar toda una serie de cuentas de usuario
nologin	Muestra un mensaje diciendo que una cuenta no está disponible; está diseñado para usarse como shell por defecto para cuentas deshabilitadas
passwd	Se usa para cambiar la contraseña de una cuenta de usuario o grupo
pwck	Verifica la integridad de los archivos de contraseñas /etc/passwd y /etc/shadow
pwconv	Crea o actualiza el archivo de contraseñas shadow a partir del archivo de contraseñas normal
pwunconv	Actualiza /etc/passwd desde /etc/shadow y luego elimina este último
sg	Ejecuta un comando dado mientras el GID del usuario está establecido al del grupo dado
su	Ejecuta un shell con IDs de usuario y grupo sustitutos
useradd	Crea un nuevo usuario con el nombre dado, o actualiza la información predeterminada de nuevos usuarios
userdel	Elimina la cuenta de usuario especificada
usermod	Se usa para modificar el nombre de inicio de sesión, UID, shell, grupo inicial, directorio personal, etc. del usuario dado
vigr	Edita los archivos `/etc/group` o `/etc/gshadow`
vipw	Edita los archivos `/etc/passwd` o `/etc/shadow`
`libsubid`	`Biblioteca para manejar rangos de ID subordinados para usuarios y grupos`

8.29. GCC-14.2.0

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

Tiempo de compilación aproximado: 46 SBU (con pruebas)
Espacio en disco requerido: 6,3 GB

8.29.1. Instalación de GCC

Si se compila en x86_64, cambie el nombre del directorio predeterminado para las bibliotecas de 64 bits a "lib":

```
case $(uname -m) in
  x86_64)
    sed -e '/m64=/s/lib64/lib/' \
        -i.orig gcc/config/i386/t-linux64
  ;;
esac
```

La documentación de GCC recomienda compilar GCC en un directorio de compilación dedicado:

```
mkdir -v build
cd       build
```

Preparar GCC para la compilación:

```
../configure --prefix=/usr             \
             LD=ld                     \
             --enable-languages=c,c++  \
             --enable-default-pie      \
             --enable-default-ssp      \
             --enable-host-pie         \
             --disable-multilib        \
             --disable-bootstrap       \
             --disable-fixincludes     \
             --with-system-zlib
```

GCC admite siete lenguajes de programación diferentes, pero los requisitos previos para la mayoría de ellos aún no se han instalado.

Consulte la página del libro BLFS de GCC para obtener instrucciones sobre cómo compilar todos los lenguajes compatibles con GCC.

Significado de los nuevos parámetros de configuración:

`LD=ld`

Este parámetro hace que el script de configuración utilice el programa ld instalado por el paquete Binutils compilado anteriormente en este capítulo, en lugar de la versión de compilación cruzada que se usaría de otro modo.

--disable-fixincludes

De forma predeterminada, durante la instalación de GCC, algunos encabezados del sistema se "corregirían" para su uso con GCC. Esto no es necesario en un sistema Linux moderno y podría ser perjudicial si se reinstala un paquete después de instalar GCC.

Esta opción impide que GCC "corrija" los encabezados.

--with-system-zlib

Esta opción indica a GCC que enlace con la copia de la biblioteca Zlib instalada en el sistema, en lugar de con su propia copia interna.

Nota

Los ejecutables independientes de la posición (PIE) son programas binarios que se pueden cargar en cualquier lugar de la memoria. Sin PIE, la función de seguridad ASLR (Aleatorización del Diseño del Espacio de Direcciones) se puede aplicar a las bibliotecas compartidas, pero no a los ejecutables. Habilitar PIE permite ASLR para los ejecutables, además de las bibliotecas compartidas, y mitiga algunos ataques basados en direcciones fijas de código o datos sensibles en los ejecutables.

SSP (Protección contra Destrución de Pila) es una técnica para garantizar que la pila de parámetros no se corrompa. La corrupción de la pila puede, por ejemplo, alterar la dirección de retorno de una subrutina, transfiriendo así el control a código peligroso (existente en el programa o en las bibliotecas compartidas, o inyectado por el atacante de alguna manera).

Compilar el paquete:

```
make
```

Importante

En esta sección, el **conjunto** de pruebas para GCC se considera importante, pero requiere mucho tiempo. Se recomienda a quienes desarrollan por primera vez que ejecuten el conjunto de pruebas. El tiempo para ejecutar las pruebas se puede reducir significativamente agregando -jx al comando make -k check a continuación, donde x es la cantidad de núcleos de CPU en su sistema.

GCC podría necesitar más espacio de pila para compilar patrones de código extremadamente complejos. Como precaución para las distribuciones de host con un límite de pila ajustado, establezca explícitamente el límite máximo del tamaño de pila en infinito. En la mayoría de las distribuciones de host (y en el sistema LFS final), el límite máximo es infinito por defecto, pero no hay problema en establecerlo explícitamente. No es necesario cambiar el límite flexible del tamaño de la pila, ya que GCC lo establecerá automáticamente en un valor apropiado, siempre que no exceda el límite rígido:

```
ulimit -s -H unlimited
```

Ahora elimine/corrija varios fallos de prueba conocidos:

```
sed -e '/cpython/d'            -i ../gcc/testsuite/gcc.dg/plugin/plugin.exp
sed -e 's/no-pic /&-no-pie /'  -i ../gcc/testsuite/gcc.target/i386/pr113689-1.c
sed -e 's/300000/(1|300000)/'  -i ../libgomp/testsuite/libgomp.c-c++-common/pr109062.c
sed -e 's/{ target nonpic } //' \
    -e '/GOTPCREL/d'           -i ../gcc/testsuite/gcc.target/i386/fentryname3.c
```

Pruebe los resultados como un usuario sin privilegios, pero no se detenga en los errores:

```
chown -R tester .
su tester -c "PATH=$PATH make -k check"
```

Para extraer un resumen de los resultados del conjunto de pruebas, ejecute:

```
../contrib/test_summary
```

Para filtrar solo los resúmenes, envíe la salida mediante **grep -A7 Summ**.

Los resultados se pueden comparar con los que se encuentran en

https://www.linuxfromscratch.org/lfs/build-logs/12.3/ y https://gcc.gnu.org/ml/gcc-testresults/

Se sabe que las pruebas tsan fallan en algunas distribuciones.

Algunos fallos inesperados no siempre se pueden evitar. En algunos casos, los fallos de las pruebas dependen del hardware específico del sistema. A menos que los resultados de la prueba sean muy diferentes a los de la URL anterior, es seguro continuar.

Instalar el paquete:

```
make install
```

El directorio de compilación de GCC ahora pertenece a tester, y la propiedad del directorio de encabezado instalado (y su contenido) es incorrecta. Cambie la propiedad al usuario y grupo root:

```
chown -v -R root:root \
    /usr/lib/gcc/$(gcc -dumpmachine)/14.2.0/include{,-fixed}
```

Cree un enlace simbólico requerido por el FHS por razones históricas.

```
ln -svr /usr/bin/cpp /usr/lib
```

Muchos paquetes usan el nombre cc para llamar al compilador de C. Ya hemos creado cc como enlace simbólico en gcc-pass2; crea también su página de manual como enlace simbólico:

```
ln -sv gcc.1 /usr/share/man/man1/cc.1
```

Añade un enlace simbólico de compatibilidad para habilitar la compilación de programas con Optimización del Tiempo de Enlace (LTO):

```
ln -sfv ../../libexec/gcc/$(gcc -dumpmachine)/14.2.0/liblto_plugin.so \
        /usr/lib/bfd-plugins/
```

Ahora que nuestra cadena de herramientas final está lista, es importante asegurar de nuevo que la compilación y el enlace funcionen correctamente. Para ello, realizamos algunas comprobaciones de seguridad:

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

No debería haber errores y la salida del último comando será (teniendo en cuenta las diferencias específicas de la plataforma en el nombre del enlazador dinámico):

```
[[Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Ahora asegúrese de que estamos configurados para usar los archivos de inicio correctos:

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

La salida del último comando debería ser:

```
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/Scrt1.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crti.o succeeded
/usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/../../../../lib/crtn.o succeeded
```

Dependiendo de la arquitectura de su equipo, lo anterior puede variar ligeramente. La diferencia radica en el nombre del directorio después de `/usr/lib/gcc`. Es importante verificar que gcc haya encontrado los tres archivos `crt*.o` en el directorio `/usr/lib`. Verifique que el compilador esté buscando los archivos de encabezado correctos:

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

Este comando debería devolver el siguiente resultado:

```
#include <...> search starts here:
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include
 /usr/local/include
 /usr/lib/gcc/x86_64-pc-linux-gnu/14.2.0/include-fixed
 /usr/include
```

Nuevamente, el directorio nombrado según el triplete de destino puede ser diferente al anterior, dependiendo de la arquitectura de su sistema.

A continuación, verifique que el nuevo enlazador se esté utilizando con las rutas de búsqueda correctas:

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

Las referencias a rutas que tengan componentes con '-linux-gnu' deben ignorarse; de lo contrario, la salida del último comando debería ser:

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib64")
SEARCH_DIR("/usr/local/lib64")
SEARCH_DIR("/lib64")
SEARCH_DIR("/usr/lib64")
```

```
SEARCH_DIR("/usr/x86_64-pc-linux-gnu/lib")
SEARCH_DIR("/usr/local/lib")
SEARCH_DIR("/lib")
SEARCH_DIR("/usr/lib");
```

Un sistema de 32 bits puede usar algunos otros directorios. Por ejemplo, aquí está la salida de una máquina i686:

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

A continuación, asegúrese de usar la libc correcta:

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

El resultado del último comando debería ser:

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

Asegúrese de que GCC esté usando el enlazador dinámico correcto:

```
grep found dummy.log
```

El resultado del último comando debería ser (teniendo en cuenta las diferencias específicas de la plataforma en el nombre del enlazador dinámico):

```
found ld-linux-x86-64.so.2 at /usr/lib/ld-linux-x86-64.so.2
```

Si el resultado no se muestra como se muestra arriba o no se recibe, entonces algo está muy mal. Investigue y vuelva a seguir los pasos para encontrar el problema y corregirlo. Cualquier problema debe resolverse antes de continuar con el proceso. Una vez que todo funcione correctamente, limpie los archivos de prueba:

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

Finalmente, mueva un archivo extraviado:

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

8.29.2. Contenido de GCC

Programas instalados: c++, cc (enlace a gcc), cpp, g++, gcc, gcc-ar, gcc-nm, gcc-ranlib, gcov, gcov-dump, gcov-tool y lto-dump

Bibliotecas instaladas: libasan.{a,so}, libatomic.{a,so}, libcc1.so, libgcc.a, libgcc_eh.a, libgcc_s.so, libgcov.a, libgomp.{a,so}, libhwasan.{a,so}, libitm.{a,so}, liblsan.{a,so}, liblto_plugin.so, libquadmath.{a,so}, libssp.{a,so}, libssp_nonshared.a, libstdc++.{a,so}, libstdc++exp.a, libstdc++fs.a, libsupc++.a, libtsan.{a,so} y libubsan.{a,so}

Directorio instalados: /usr/include/c++, /usr/lib/gcc, /usr/libexec/gcc y /usr/share/gcc-14.2.0

Descripciones breves

c++ El compilador de C++

cc El compilador de C

cpp El preprocesador de C; El compilador lo utiliza para expandir las directivas #include, #define y similares en los archivos fuente

g++ El compilador de C++

gcc El compilador de C

gcc-ar Un contenedor de ar que añade un complemento a la línea de comandos. Este programa solo se utiliza para añadir "optimización del tiempo de enlace" y no es útil con las opciones de compilación predeterminadas

gcc-nm Un contenedor de nm que añade un complemento a la línea de comandos. Este programa solo se utiliza para añadir "optimización del tiempo de enlace" y no es útil con las opciones de compilación predeterminadas.

gcc-ranlib Un contenedor de ranlib que añade un complemento a la línea de comandos Este programa solo se utiliza para añadir "optimización del tiempo de enlace" y no es útil con las opciones de compilación predeterminadas

gcov Una herramienta de pruebas de cobertura. Se utiliza para analizar programas y determinar dónde las optimizaciones tendrán el mayor efecto

gcov-dump Herramienta de volcado de perfiles gcda y gcno sin conexión

gcov-tool Herramienta de procesamiento de perfiles gcda sin conexión.

lto-dump Herramienta para volcar archivos objeto generados por GCC con LTO habilitado

libasan Biblioteca de ejecución Address Sanitizer.

libatomic Biblioteca de ejecución atómica integrada de GCC.

libcc1 Una biblioteca que permite a GDB utilizar GCC.

libgcc Contiene soporte en tiempo de ejecución para gcc.

libgcov	Esta biblioteca se vincula a un programa cuando se le indica a GCC que habilite la generación de perfiles.
libgomp	Implementación GNU de la API OpenMP para programación paralela multiplataforma de memoria compartida en C/C++ y Fortran.
libhwasan	Biblioteca de ejecución Address Sanitizer asistida por hardware
libitm	Biblioteca de memoria transaccional de GNU
liblsan	Biblioteca de ejecución Leak Sanitizer
liblto_plugin	El complemento LTO de GCC permite a Binutils procesar archivos objeto generados por GCC con LTO habilitado
libquadmath	API de la biblioteca matemática de precisión cuádruple de GCC
libssp	Contiene rutinas que soportan la función de protección contra la destrucción de pila de GCC. Normalmente no se utiliza, ya que Glibc también proporciona dichas rutinas
libstdc++	La biblioteca estándar de C++
libstdc++exp	La biblioteca experimental de contratos de C++
libstdc++fs	La biblioteca del sistema de archivos ISO/IEC TS 18822:2015
libsupc++	Proporciona rutinas de soporte para el lenguaje de programación C++
libtsan	La biblioteca de ejecución Thread Sanitizer
libubsan	La biblioteca de ejecución Undefined Behavior Sanitizer

8.30. Ncurses-6.5

El paquete Ncurses contiene bibliotecas para la gestión de pantallas de caracteres independiente de la terminal.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 46 MB

8.30.1. Instalación de Ncurses

Preparar Ncurses para la compilación:

```
./configure --prefix=/usr           \
            --mandir=/usr/share/man \
            --with-shared           \
            --without-debug         \
            --without-normal        \
            --with-cxx-shared       \
            --enable-pc-files       \
            --with-pkg-config-libdir=/usr/lib/pkgconfig
```

Significado de las nuevas opciones de configuración:

`--with-shared`

Esto hace que Ncurses compile e instale bibliotecas de C compartidas.

`--without-normal`

Esto impide que Ncurses compile e instale bibliotecas de C estáticas.

`--without-debug`

Esto impide que Ncurses compile e instale bibliotecas de depuración.

`--with-cxx-shared`

Esto hace que Ncurses compile e instale enlaces compartidos de C++. También impide que compile e instale enlaces estáticos de C++.

`--enable-pc-files`

Esta opción genera e instala archivos .pc para pkg-config.

Compilar el paquete:

```
make
```

Este paquete incluye un conjunto de pruebas, pero solo se puede ejecutar después de instalarlo. Las pruebas se encuentran en el directorio test/. Consulte el archivo README de ese directorio para obtener más información.

La instalación de este paquete sobrescribirá `libncursesw.so.6.5` localmente. Esto podría bloquear el proceso del shell que utiliza el código y los datos del archivo de la biblioteca. Instale el paquete con `DESTDIR` y reemplace el archivo de la biblioteca correctamente con el comando **install** (el encabezado `curses.h` también se edita para garantizar que se use la ABI de caracteres anchos, como se hizo en la Sección 6.3, “Ncurses-6.5”):

```
make DESTDIR=$PWD/dest install
install -vm755 dest/usr/lib/libncursesw.so.6.5 /usr/lib
rm -v dest/usr/lib/libncursesw.so.6.5
sed -e 's/^#if.*XOPEN.*$/#if 1/' \
    -i dest/usr/include/curses.h
cp -av dest/* /
```

Muchas aplicaciones aún esperan que el enlazador pueda encontrar bibliotecas de Ncurses que no sean de caracteres anchos. Engañe a estas aplicaciones para que enlacen con bibliotecas de caracteres anchos mediante enlaces simbólicos (tenga en cuenta que los enlaces .so solo son seguros con curses.h). Editado para usar siempre la ABI de caracteres anchos:

```
for lib in ncurses form panel menu ; do
    ln -sfv lib${lib}w.so /usr/lib/lib${lib}.so
    ln -sfv ${lib}w.pc    /usr/lib/pkgconfig/${lib}.pc
done
```

Finalmente, asegúrese de que las aplicaciones antiguas que buscan "-lcurses" en tiempo de compilación aún se puedan compilar:

```
ln -sfv libncursesw.so /usr/lib/libcurses.so
```

Si lo desea, instale la documentación de Ncurses:

```
cp -v -R doc -T /usr/share/doc/ncurses-6.5
```

Nota

Las instrucciones anteriores no crean bibliotecas de Ncurses que no sean de caracteres anchos, ya que ningún paquete instalado al compilar desde el código fuente las enlazaría en tiempo de ejecución. Sin embargo, las únicas aplicaciones **conocidas** de solo binario que enlazan con bibliotecas Ncurses de caracteres no anchos requieren la versión 5. Si necesita tener dichas bibliotecas debido a alguna aplicación de solo **binario** o para cumplir con LSB, vuelva a compilar el paquete con los siguientes comandos:

```
make distclean
./configure --prefix=/usr         \
            --with-shared         \
            --without-normal      \
            --without-debug       \
            --without-cxx-binding \
            --with-abi-version=5
make sources libs
cp -av lib/lib*.so.5* /usr/lib
```

8.30.2. Contenido de Ncurses

Programas instalados: captoinfo (enlace a tic), clear, infocmp, infotocap (enlace a tic), ncursesw6-config, reset (enlace a tset), tabs, tic, toe, tput y tset

Bibliotecas instaladas: libcurses.so (enlace simbólico), libform.so (enlace simbólico), libformw.so, libmenu.so (enlace simbólico), libmenuw.so, libncurses.so (enlace simbólico), libncursesw.so, libncurses++w.so, libpanel.so (enlace simbólico) y libpanelw.so,

Directorios instalados: /usr/share/tabset, /usr/share/terminfo y /usr/share/doc/ncurses-6.5

Descripciones breves

captoinfo	Convierte una descripción de termcap en una descripción de terminfo
clear	Limpia la pantalla, si es posible
infocmp	Compara o imprime descripciones de terminfo
infotocap	Convierte una descripción de terminfo en una descripción de termcap
ncursesw6-config	Proporciona información de configuración para ncurses
reset	Reinicializa una terminal a sus valores predeterminados
tabs	Limpia y establece tabulaciones en una terminal
tic	El compilador de descripción de entrada de terminfo que traduce un archivo terminfo del formato fuente al formato binario necesario para las rutinas de la biblioteca ncurses [Un archivo terminfo contiene información sobre las capacidades de una terminal específica]
toe	Enumera todos los tipos de terminal disponibles, indicando el nombre principal y la descripción de cada uno
tput	Pone a disposición del shell los valores de las capacidades dependientes de la terminal. También se puede usar para reiniciar o inicializar una terminal o informar su nombre completo
tset	Se puede usar para inicializar terminales
`libncursesw`	Contiene funciones para mostrar texto de muchas formas complejas en una pantalla de terminal; un buen ejemplo del uso de estas funciones es el menú que se muestra durante el **make menuconfig** del kernel.
`libncurses++w`	`Contiene enlaces de C++ para otras bibliotecas de este paquete`
`libformw`	`Contiene funciones para implementar formularios`
`libmenuw`	`Contiene funciones para implementar menús`
`libpanelw`	`Contiene funciones para implementar paneles`

8.31. Sed-4.9

El paquete Sed contiene un editor de flujos.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 30 MB

8.31.1. Instalación de Sed

Preparar Sed para la compilación:

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

Compilar el paquete y generar la documentación HTML:

```
make
make html
```

Para probar los resultados, ejecute:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Instalar el paquete y su documentación:

```
make install
install -d -m755           /usr/share/doc/sed-4.9
install -m644 doc/sed.html /usr/share/doc/sed-4.9
```

8.31.2. Contenido de Sed

Programa instalado: sed
Directorio de instalación: /usr/share/doc/sed-4.9

Descripciones breves

sed Filtra y transforma archivos de texto en una sola pasada

8.32. Psmisc-23.7

El paquete Psmisc contiene programas para mostrar información sobre los procesos en ejecución.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 6,7 MB

8.32.1. Instalación de Psmisc

Preparar Psmisc para la compilación:

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

Compilar el paquete:

```
make
```

Para ejecutar el conjunto de pruebas, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.32.2. Contenido de Psmisc

Programas instalados: fuser, killall, peekfd, prtstat, pslog, pstree y pstree.x11 (enlace a pstree)

Descripciones breves

fuser Informa de los identificadores de proceso (PID) de los procesos que utilizan los archivos o sistemas de archivos especificados.

killall Detiene los procesos por nombre. Envía una señal a todos los procesos que ejecutan cualquiera de los comandos dados.

peekfd Examina los descriptores de archivo de un proceso en ejecución, dado su PID.

prtstat Imprime información sobre un proceso.

pslog Informa sobre la ruta actual de los registros de un proceso.

pstree Muestra los procesos en ejecución como un árbol.

pstree.x11 Igual que **pstree**, excepto que espera confirmación antes de salir.

8.33. Gettext-0.24

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

Tiempo de compilación aproximado: 1.7 SBU
Espacio en disco requerido: 290 MB

8.33.1. Instalación de Gettext

Preparar Gettext para la compilación:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/gettext-0.24
```

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

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

8.33.2. Contenido de Gettext

Programas instalados: autopoint, envsubst, gettext, gettext.sh, gettextize, msgattrib, msgcat, msgcmp, msgcomm, msgconv, msgen, msgexec, msgfilter, msgfmt, msggrep, msginit, msgmerge, msgunfmt, msguniq, ngettext, recode-sr-latin y xgettext

Bibliotecas instaladas: libasprintf.so, libgettextlib.so, libgettextpo.so, libgettextsrc.so, libtextstyle.so y preloadable_libintl.so

Directorios instalados: /usr/lib/gettext, /usr/share/doc/gettext-0.24, /usr/share/gettext y /usr/share/gettext-0.24

Descripciones breves

autopoint Copia los archivos estándar de la infraestructura Gettext en un paquete fuente.

envsubst Sustituye las variables de entorno en cadenas de formato de shell.

gettext	Traduce un mensaje en lenguaje natural al idioma del usuario consultando la traducción en un catálogo de mensajes.
gettext.sh	Sirve principalmente como biblioteca de funciones de shell para gettext.
gettextize	Copia todos los archivos estándar Gettext en el directorio de nivel superior de un paquete para comenzar a internacionalizarlo.
msgattrib	Filtra los mensajes de un catálogo de traducción según sus atributos y los manipula.
msgcat	Concatena y fusiona los archivos .po.
msgcmp	Compara dos archivos .po para comprobar que ambos contienen el mismo conjunto de cadenas msgid.
msgcomm	Busca los mensajes comunes a los archivos .po especificados.
msgconv	Convierte un catálogo de traducción a una codificación de caracteres diferente.
msgen	Crea un catálogo de traducción al inglés.
msgexec	Aplica un comando a todas las traducciones de un catálogo de traducción.
msgfilter	Aplica un filtro a todas las traducciones de un catálogo de traducción.
msgfmt	Genera un catálogo de mensajes binarios a partir de un catálogo de traducción.
msggrep	Extrae todos los mensajes de un catálogo de traducción que coinciden con un patrón determinado o pertenecen a algunos archivos fuente.
msginit	Crea un nuevo archivo .po, inicializando la metainformación con valores del entorno del usuario.
msgmerge	Combina. Dos traducciones sin procesar en un solo archivo
msgunfmt	Descompila un catálogo de mensajes binarios en texto de traducción sin procesar
msguniq	Unifica traducciones duplicadas en un catálogo de traducción
ngettext	Muestra traducciones al idioma nativo de un mensaje de texto cuya forma gramatical depende de un número
recode-sr-latin	Recodifica texto serbio del alfabeto cirílico al latino
xgettext	Extrae las líneas de mensaje traducibles de los archivos fuente dados para crear la primera plantilla de traducción
libasprintf	Define la clase autosprintf, que permite usar rutinas de salida con formato C en programas C++, para su uso con las cadenas <string> y los flujos <iostream>
libgettextlib	Contiene rutinas comunes utilizadas por los diversos programas Gettext; estas no están diseñadas para uso general

libgettextpo Se utiliza para escribir programas especializados que procesan archivos .po; esta biblioteca se utiliza cuando las aplicaciones estándar incluidas con Gettext (como msgcomm, msgcmp, msgattrib y msgen) no son suficientes

libgettextsrc Proporciona rutinas comunes utilizadas por los diversos programas Gettext; Estos no están destinados a uso general.

libtextstyle Biblioteca de estilos de texto.

preloadable_libintl Una biblioteca, diseñada para ser utilizada por LD_PRELOAD, que ayuda a libintl a registrar mensajes no traducidos.

8.34. Bison-3.8.2

El paquete Bison contiene un generador de analizadores sintácticos.

Tiempo de compilación aproximado: 2.1 SBU
Espacio en disco requerido: 62 MB

8.34.1. Instalación de Bison

Preparar Bison para la compilación:

```
./configure --prefix=/usr –docdir=/usr/share/doc/bison-3.8.2
```

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.34.2. Contenido de Bison

Programas instalados: bison y yacc
Biblioteca instalada: liby.a
Directorio de instalación: /usr/share/bison

Descripciones breves

bison Genera, a partir de una serie de reglas, un programa para analizar la estructura de archivos de texto. Bison es un reemplazo de Yacc (Yet Another Compiler Compiler).

yacc Un contenedor para bison, pensado para programas que aún llaman a yacc en lugar de bison; llama a bison con la opción -y.

liby La biblioteca de Yacc que contiene implementaciones de las funciones yyerror y main compatibles con Yacc. Esta biblioteca no suele ser muy útil, pero POSIX la requiere.

8.35. Grep-3.11

El paquete Grep contiene programas para buscar en el contenido de los archivos.

Tiempo de compilación aproximado: 0.4 SBU
Espacio en disco requerido: 39 MB

8.35.1. Instalación de Grep

Primero, elimine una advertencia sobre el uso de egrep y fgrep que provoca el fallo de las pruebas en algunos paquetes:

```
sed -i "s/echo/#echo/" src/egrep.sh
```

Prepare Grep para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.35.2. Contenido de Grep

Programas instalados: egrep, fgrep y grep

Descripciones breves

egrep Imprime las líneas que coinciden con una expresión regular extendida. Está obsoleto, use grep -E en su lugar.

fgrep Imprime líneas que coinciden con una lista de cadenas fijas. Está obsoleto, use grep -F en su lugar.

grep Imprime líneas que coinciden con una expresión regular básica.

8.36. Bash-5.2.37

El paquete Bash contiene Bourne-Again Shell.

Tiempo de compilación aproximado: 1.4 SBU
Espacio en disco requerido: 53 MB

8.36.1. Instalación de Bash

Preparar Bash para la compilación:

```
./configure --prefix=/usr             \
            --without-bash-malloc     \
            --with-installed-readline \
            --docdir=/usr/share/doc/bash-5.2.37
```

Significado de la nueva opción de configuración:

`--with-installed-readline`

Esta opción indica a Bash que use la biblioteca readline ya instalada en el sistema en lugar de usar su propia versión de readline.

Compilar el paquete:

```
make
```

Vaya a "Instalar el paquete" si no está ejecutando el conjunto de pruebas.

Para preparar las pruebas, asegúrese de que el usuario `tester` pueda escribir en el árbol de fuentes:

```
chown -R tester .
```

El conjunto de pruebas de este paquete está diseñado para ejecutarse como un usuario no-`root`, propietario de la terminal conectada a la entrada estándar. Para cumplir con este requisito, cree una nueva pseudoterminal con Expect y ejecute las pruebas como el usuario `tester`:

```
su -s /usr/bin/expect tester << "EOF"
set timeout -1
spawn make tests
expect eof
lassign [wait] _ _ _ value
exit $value
EOF
```

El conjunto de pruebas utiliza **diff** para detectar la diferencia entre la salida del script de prueba y la salida esperada. Cualquier salida de **diff** (con los prefijos < y >) indica un fallo en la prueba, a menos que aparezca un mensaje indicando que la diferencia puede ignorarse. Se sabe que una prueba llamada `run-builtins` falla en algunas distribuciones de host con una diferencia en la primera línea de la salida.

Instale el paquete:

```
make install
```

Ejecute el programa bash recién compilado (reemplazando el que se está ejecutando):

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

8.36.2. Contenido de Bash

Programas instalados: bash, bashbug y sh (enlace a bash)

Directorio de instalación: /usr/include/bash, /usr/lib/bash y /usr/share/doc/bash-5.2.37

Descripciones breves

bash Un intérprete de comandos ampliamente utilizado; realiza diversos tipos de expansiones y sustituciones en una línea de comandos antes de ejecutarla, lo que lo convierte en una herramienta poderosa.

bashbug Un script de shell que ayuda al usuario a redactar y enviar por correo informes de errores con formato estándar relacionados con bash.

sh Un enlace simbólico al programa bash; cuando se invoca como sh, bash intenta imitar el comportamiento de inicio de versiones anteriores de sh con la mayor fidelidad posible, a la vez que cumple con el estándar POSIX.

8.37. Libtool-2.5.4

El paquete Libtool contiene el script de soporte para bibliotecas genéricas de GNU. Simplifica el uso de bibliotecas compartidas con una interfaz consistente y portátil.

Tiempo de compilación aproximado: 0.6 SBU
Espacio en disco requerido: 44 MB

8.37.1. Instalación de Libtool

Preparar Libtool para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Eliminar una biblioteca estática inútil:

```
rm -fv /usr/lib/libltdl.a
```

8.37.2. Contenido de Libtool

Programas instalados: libtool y libtoolize
Bibliotecas instaladas: libltdl.so
Directorios instalados: /usr/include/libltdl y /usr/share/libtool

Descripciones breves

libtool Proporciona servicios generalizados de soporte para la creación de bibliotecas.

libtoolize Proporciona una forma estándar de añadir soporte para **libtool** a un paquete.

`libltdl` Oculta las diversas dificultades al abrir bibliotecas cargadas dinámicamente.

8.38. GDBM-1.24

El paquete GDBM contiene el Gestor de Bases de Datos GNU. Es una biblioteca de funciones de base de datos que utiliza hash extensible y funciona como el dbm estándar de UNIX. La biblioteca proporciona primitivas para almacenar pares clave/datos, buscar y recuperar datos por su clave, y eliminar una clave junto con sus datos.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 13 MB

8.38.1. Instalación de GDBM

Preparar GDBM para la compilación:

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

Significado de la opción de configuración:

`--enable-libgdbm-compat`

Esta opción permite compilar la biblioteca de compatibilidad libgdbm. Algunos paquetes fuera de LFS pueden requerir las rutinas DBM más antiguas que proporciona.

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.38.2. Contenido de GDBM

Programas instalados: gdbm_dump, gdbm_load y gdbmtool
Bibliotecas instaladas: libgdbm.so y libgdbm_compat.so

Descripciones breves

gdbm_dump Vuelca una base de datos de GDBM a un archivo

gdbm_load Recrea una base de datos de GDBM a partir de un archivo de volcado

gdbmtool Prueba y modifica una base de datos de GDBM

`libgdbm` Contiene funciones para manipular una base de datos con hash

`libgdbm_compat` Biblioteca de compatibilidad con funciones de DBM antiguas

8.39. Gperf-3.1

Gperf genera una función hash perfecta a partir de un conjunto de claves.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 6,1 MB

8.39.1. Instalación de Gperf

Preparar Gperf para la compilación:

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

Compilar el paquete:

```
make
```

Se sabe que las pruebas fallan si se ejecutan varias simultáneamente (opción -j mayor que 1). Para comprobar los resultados, ejecute:

```
make -j1 check
```

Instalar el paquete:

```
make install
```

8.39.2. Contenido de Gperf

Programa instalado: gperf
Directorio de instalación: /usr/share/doc/gperf-3.1

Descripciones breves

gperf Genera un hash perfecto a partir de un conjunto de claves

8.40. Expat-2.6.4

El paquete Expat contiene una biblioteca de C orientada a flujos para analizar XML.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 14 MB

8.40.1. Instalación de Expat

Preparar Expat para la compilación:

```
./configure --prefix=/usr    \
            --disable-static \
            --docdir=/usr/share/doc/expat-2.6.4
```

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Si lo desea, instale la documentación:

```
install -v -m644 doc/*.{html,css} /usr/share/doc/expat-2.6.4
```

8.40.2. Contenido de Expat

Programa instalado: xmlwf
Bibliotecas instaladas: libexpat.so
Directorio de instalación: /usr/share/doc/expat-2.6.4

Descripciones breves

xmlwf Es una utilidad sin validación que comprueba si los documentos XML están correctamente formados.

libexpat Contiene funciones API para analizar XML.

8.41. Inetutils-2.6

El paquete Inetutils contiene programas para redes básicas.

Tiempo de compilación aproximado: 0,2 SBU
Espacio en disco requerido: 35 MB

8.41.1. Instalación de Inetutils

Primero, compile el paquete con gcc-14.1 o posterior:

```
sed -i 's/def HAVE_TERMCAP_TGETENT/ 1/' telnet/telnet.c
```

Prepare Inetutils para la compilación:

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

Significado de las opciones de configuración:

`--disable-logger`

Esta opción impide que Inetutils instale el programa de **registro**, que los scripts utilizan para enviar mensajes al demonio de registro del sistema. No lo instale, ya que Util-linux instala una versión más reciente.

`--disable-whois`

Esta opción deshabilita la creación del cliente **whois** de Inetutils, que está desactualizado. Las instrucciones para un mejor cliente **whois** se encuentran en el manual de BLFS.

`--disable-r*`

Estos parámetros deshabilitan la creación de programas obsoletos que no deben usarse debido a problemas de seguridad. Las funciones proporcionadas por estos programas pueden proporcionarse mediante el paquete openssh del manual de BLFS.

`--disable-servers`

Esto deshabilita la instalación de los diversos servidores de red incluidos en el paquete Inetutils. Estos servidores no se consideran adecuados para un sistema LFS básico. Algunos son inseguros por naturaleza y solo se consideran seguros en redes confiables. Tenga en cuenta que existen alternativas mejores para muchos de estos servidores.

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Mueva un programa a la ubicación correcta:

```
mv -v /usr/{,s}bin/ifconfig
```

8.42. Less-668

El paquete Less contiene un visor de archivos de texto.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 14 MB

8.42.1. Instalación de Less

Preparar Less para la compilación:

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

Significado de las opciones de configuración:

`--sysconfdir=/etc`

Esta opción indica a los programas creados por el paquete que busquen los archivos de configuración en `/etc`.

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.42.2. Contenido de Less

Programas instalados: less, lessecho y lesskey

Descripciones breves

less Un visor o paginador de archivos; Muestra el contenido del archivo dado, permitiendo al usuario desplazarse, encontrar cadenas y acceder a las marcas.

lessecho Necesario para expandir metacaracteres, como * y ?, en nombres de archivo en sistemas Unix.

lesskey Se utiliza para especificar las combinaciones de teclas para **less**.

8.43. Perl-5.40.1

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

Tiempo de compilación aproximado: 1.3 SBU
Espacio en disco requerido: 245 MB

8.43.1. Instalación de Perl

Esta versión de Perl compila los módulos Compress::Raw::Zlib y Compress::Raw::BZip2. Por defecto, Perl utilizará una copia interna del código fuente para la compilación. Ejecute el siguiente comando para que Perl utilice las bibliotecas instaladas en el sistema:

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

Para tener control total sobre la configuración de Perl, puede eliminar las opciones "-des" del siguiente comando y configurar manualmente la compilación de este paquete. Como alternativa, use el comando exactamente como se muestra a continuación para usar los valores predeterminados que Perl detecta automáticamente:

```
sh Configure -des                                         \
             -D prefix=/usr                               \
             -D vendorprefix=/usr                         \
             -D privlib=/usr/lib/perl5/5.40/core_perl     \
             -D archlib=/usr/lib/perl5/5.40/core_perl     \
             -D sitelib=/usr/lib/perl5/5.40/site_perl     \
             -D sitearch=/usr/lib/perl5/5.40/site_perl    \
             -D vendorlib=/usr/lib/perl5/5.40/vendor_perl  \
             -D vendorarch=/usr/lib/perl5/5.40/vendor_perl \
             -D man1dir=/usr/share/man/man1               \
             -D man3dir=/usr/share/man/man3               \
             -D pager="/usr/bin/less -isR"                \
             -D useshrplib                                \
             -D usethreads
```

Significado de las nuevas opciones de Configure:

-D pager="/usr/bin/less -isR"

Esto garantiza que se use **less** en lugar de **more**.

-D man1dir=/usr/share/man/man1 -D man3dir=/usr/share/man/man3

Dado que Groff aún no está instalado, **Configure** no creará páginas de manual para Perl. Estos parámetros anulan este comportamiento.

-D usethreads

Compile Perl con soporte para hilos.

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
TEST_JOBS=$(nproc) make test_harness
```

Instalar el paquete y limpiarlo:

```
make install
unset BUILD_ZLIB BUILD_BZIP2
```

8.43.2. Contenido de Perl

Programas instalados: corelist, cpan, enc2xs, encguess, h2ph, h2xs, instmodsh, json_pp, libnetcfg, perl, perl5.40.1 (enlace directo a perl), perlbug, perldoc, perlivp, perlthanks (enlace directo a perlbug), piconv, pl2pm, pod2html, pod2man, pod2text, pod2usage, podchecker, podselect, prove, ptar, ptardiff, ptargrep, shasum, splain, xsubpp y zipdetails

Bibliotecas instaladas: Muchas de las cuales no se pueden enumerar todas aquí

Directorio de instalación: /usr/lib/perl5

Descripciones breves

corelist Una interfaz de línea de comandos para Module::CoreList

cpan Interactúa con la Red Integral de Archivos de Perl (CPAN) desde la línea de comandos

enc2xs Crea una extensión de Perl para el módulo Encode a partir de asignaciones de caracteres Unicode o archivos de codificación Tcl

encguess Adivina el tipo de codificación de uno o varios archivos

h2ph Convierte archivos de encabezado .h de C a archivos de encabezado .ph de Perl

h2xs Convierte archivos de encabezado .h de C a extensiones de Perl

instmodsh Script de shell Para examinar los módulos Perl instalados; puede crear un archivo tar a partir de un módulo instalado.

json_pp Convierte datos entre ciertos formatos de entrada y salida.

libnetcfg Puede usarse para configurar el módulo Perl libnet.

perl Combina algunas de las mejores características de C, sed, awk y sh en un único lenguaje de programación.

perl5.40.1 Un enlace directo a Perl.

perlbug Se usa para generar informes de errores sobre Perl o los módulos que lo acompañan y enviarlos por correo.

perldoc Muestra documentación en formato pod, incrustada en el árbol de instalación de Perl o en un script de Perl.

perlivp El procedimiento de verificación de la instalación de Perl. Se puede usar para verificar que Perl y sus bibliotecas se hayan instalado correctamente.

perlthanks Se usa para generar mensajes de agradecimiento para los desarrolladores de Perl.

piconv Una versión en Perl del convertidor de codificación de caracteres iconv.

pl2pm Una herramienta básica para convertir archivos .pl de Perl4 a módulos .pm de Perl5.

pod2html	Convierte archivos de formato pod a formato HTML.
pod2man	Convierte datos de pod a entrada *roff formateada.
pod2text	Convierte datos de pod a texto ASCII formateado.
pod2usage	Imprime mensajes de uso de la documentación de pod incrustada en archivos.
podchecker	Comprueba la sintaxis de los archivos de documentación en formato pod.
podselect	Muestra secciones seleccionadas de la documentación de pod.
prove	Herramienta de línea de comandos para ejecutar pruebas con el módulo Test::Harness.
ptar	Un programa similar a tar escrito en Perl.
ptardiff	Un programa en Perl que compara un archivo extraído con uno sin extraer.
ptargrep	Un programa en Perl que aplica la coincidencia de patrones al contenido de los archivos en un archivo tar.
shasum	Imprime o comprueba sumas de comprobación SHA.
splain	Se usa para forzar diagnósticos de advertencia detallados en Perl.
xsubpp	Convierte el código Perl XS en código C
zipdetails	Muestra detalles sobre la estructura interna de un archivo Zip

8.44. XML::Parser-2.47

El módulo XML::Parser es una interfaz de Perl para el analizador XML Expat de James Clark.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 2,4 MB

8.44.1. Instalación de XML::Parser

Preparar XML::Parser para la compilación:

```
perl Makefile.PL
```

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make test
```

Instalar el paquete:

```
make install
```

8.44.2. Contenido de XML::Parser

Módulo instalado: Expat.so

Descripciones breves

`Expat` proporciona la interfaz Perl Expat

8.45. Intltool-0.51.0

Intltool es una herramienta de internacionalización que se utiliza para extraer cadenas traducibles de archivos fuente.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 1,5 MB

8.45.1. Instalación de Intltool

Primero, corrija una advertencia causada por Perl-5.22 y versiones posteriores:

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

Nota

La expresión regular anterior parece inusual debido a las barras invertidas. Lo que hace es agregar una barra **invertida** antes de la llave derecha en la secuencia '\${', lo que resulta en '\$\{'.

Preparar Intltool para la compilación:

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

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

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

8.45.2. Contenido de Intltool

Programas instalados: intltool-extract, intltool-merge, intltool-prepare, intltool-update e intltoolize
Directorios instalados: /usr/share/doc/intltool-0.51.0 y /usr/share/intltool

Descripciones breves

Intltoolize Prepara un paquete para usar intltool

intltool-extract Genera archivos de encabezado que gettext puede leer

i**ntltool-merge** Fusiona cadenas traducidas en varios tipos de archivo

intltool-prepare Actualiza los archivos pot y los fusiona con los archivos de traducción

intltool-update Actualiza los archivos de plantilla po y los fusiona con las traducciones

8.46. Autoconf-2.72

El paquete Autoconf contiene programas para generar scripts de shell que configuran automáticamente el código fuente.

Tiempo de compilación aproximado: Menos de 0,1 SBU (aproximadamente 0,4 SBU con pruebas)
Espacio en disco requerido: 25 MB

8.46.1. Instalación de Autoconf

Preparar Autoconf para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.46.2. Contenido de Autoconf

Programas instalados: autoconf, autoheader, autom4te, autoreconf, autoscan, autoupdate e ifnames
Directorio de instalación: /usr/share/autoconf

Descripciones breves

autoconf Genera scripts de shell que configuran automáticamente los paquetes de código fuente de software para adaptarse a diversos tipos de sistemas tipo Unix. Los scripts de configuración que produce son independientes; su ejecución no requiere el programa autoconf.

autoheader Una herramienta para crear archivos de plantilla de sentencias C#define para que configure los use.

autom4te Un contenedor para el procesador de macros M4.

autoreconf Ejecuta automáticamente **autoconf**, **autoheader**, **aclocal**, **automake**, **gettextize** y **libtoolize** en el orden correcto para ahorrar tiempo al realizar cambios en los archivos de plantilla de **autoconf** y **automake**.

autoscan Ayuda a crear un archivo configure.in para un paquete de software. Examina los archivos fuente en un árbol de directorios, buscando problemas comunes de portabilidad,

y crea un archivo `configure.scan` que sirve como archivo `configure.in` preliminar para el paquete.

autoupdate Modifica un archivo `configure.in` que aún llama a las macros de **autoconf** por sus nombres antiguos para usar los nombres de macro actuales.

ifnames Ayuda al escribir archivos `configure.in` para un paquete de software; imprime los identificadores que el paquete usa en los condicionales del preprocesador de C. [Si un paquete ya se ha configurado para tener cierta portabilidad, este programa puede ayudar a determinar qué debe revisar **configure**. También puede completar las lagunas en un archivo `configure.in` generado por **autoescaneo**].

8.47. Automake-1.17

El paquete Automake contiene programas para generar archivos Makefile para usar con Autoconf.

Tiempo de compilación aproximado: Menos de 0.1 SBU (aproximadamente 1.1 SBU con pruebas)
Espacio en disco requerido: 121 MB

8.47.1. Instalación de Automake

Preparar Automake para la compilación:

```
./configure --prefix=/usr --docdir=/usr/share/doc/automake-1.17
```

Compilar el paquete:

```
make
```

Usar cuatro trabajos en paralelo acelera las pruebas, incluso en sistemas con menos núcleos lógicos, debido a retrasos internos en las pruebas individuales. Para comprobar los resultados, ejecute:

```
make -j$(($(nproc)>4?$(nproc):4)) check
```

Reemplace *$((...))* con el número de núcleos lógicos que desea usar si no desea usarlos todos.

Instalar el paquete:

```
make install
```

8.47.2. Contenido de Automake

Programas instalados: aclocal, aclocal-1.17 (vinculado directamente con aclocal), automake y automake-1.17 (vinculado directamente con automake)

Directorios instalados: /usr/share/aclocal-1.17, /usr/share/automake-1.17 y /usr/share/doc/automake-1.17

Descripciones breves

aclocal Genera archivos `aclocal.m4` basándose en el contenido de los archivos `configure.in`.

aclocal-1.17 Un enlace directo a **aclocal**.

automake Una herramienta para generar automáticamente archivos `Makefile.in` a partir de archivos `Makefile.am`. [Para crear todos los archivos Makefile.in de un paquete, ejecute este programa en el directorio de nivel superior. Al escanear el archivo `configure.in`, encuentra automáticamente cada archivo Makefile.am apropiado y genera el archivo `Makefile.in` correspondiente.]

automake-1.17 Un enlace directo a **automake**

8.48. OpenSSL-3.4.1

El paquete OpenSSL contiene herramientas de gestión y bibliotecas relacionadas con la criptografía. Estas son útiles para proporcionar funciones criptográficas a otros paquetes, como OpenSSH, aplicaciones de correo electrónico y navegadores web (para acceder a sitios HTTPS).

Tiempo de compilación aproximado: 1.8 SBU
Espacio en disco requerido: 920 MB

8.48.1. Instalación de OpenSSL

Preparar OpenSSL para la compilación:

```
./config --prefix=/usr         \
         --openssldir=/etc/ssl \
         --libdir=lib          \
         shared                \
         zlib-dynamic
```

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
HARNESS_JOBS=$(nproc) make test
```

Se sabe que una prueba, 30-test_afalg.t, falla si el kernel del host no tiene habilitado `CONFIG_CRYPTO_USER_API_SKCIPHER` o no tiene habilitada ninguna opción que proporcione un AES con implementación CBC (por ejemplo, la combinación de `CONFIG_CRYPTO_AES` y `CONFIG_CRYPTO_CBC`, o `CONFIG_CRYPTO_AES_NI_INTEL` si la CPU admite AES-NI). Si falla, se puede ignorar sin problemas.

Instalar el paquete:

```
sed -i '/INSTALL_LIBS/s/libcrypto.a libssl.a//' Makefile
make MANSUFFIX=ssl install
```

Añadir la versión al nombre del directorio de documentación para mantener la coherencia con otros paquetes:

```
mv -v /usr/share/doc/openssl /usr/share/doc/openssl-3.4.1
```

Si se desea, instalar documentación adicional:

```
cp -vfr doc/* /usr/share/doc/openssl-3.4.1
```

Nota

Debe actualizar OpenSSL cuando se anuncie una nueva versión que corrija vulnerabilidades. Desde OpenSSL 3.0.0, el esquema de versiones de OpenSSL sigue el formato MAJOR.MINOR.PATCH. La compatibilidad con API/ABI está garantizada para el mismo número de versión MAJOR. Dado que LFS solo instala las bibliotecas compartidas, no es necesario recompilar los paquetes que **enlazan** con libcrypto.so o libssl.so al actualizar a una versión con el mismo número de versión PRINCIPAL. Sin embargo, cualquier programa en ejecución vinculado a esas bibliotecas debe detenerse y reiniciarse. Para más información, consulte las entradas relacionadas en la Sección 8.2.1, "Problemas de actualización".

8.48.2. Contenido de OpenSSL

Programas instalados: c_rehash y openssl
Bibliotecas instaladas: libcrypto.so y libssl.so
Directorios instalados: /etc/ssl, /usr/include/openssl, /usr/lib/engines y /usr/share/doc/openssl-3.4.1

Descripciones breves

c_rehash es un script de Perl que escanea todos los archivos de un directorio y añade enlaces simbólicos a sus valores hash. El uso de **c_rehash** se considera obsoleto y debería reemplazarse por el comando **openssl rehash**.

openssl es una herramienta de línea de comandos para usar las diversas funciones criptográficas de la biblioteca de cifrado de OpenSSL desde el shell. Se puede utilizar para diversas funciones documentadas en *openssl(1)*.

`libcrypto.so` implementa una amplia gama de algoritmos criptográficos utilizados en diversos estándares de Internet. Los servicios que ofrece esta biblioteca son utilizados por las implementaciones de OpenSSL de SSL, TLS y S/MIME, y también se han empleado para implementar OpenSSH, OpenPGP y otros estándares criptográficos.

`libssl.so` implementa el protocolo de Seguridad de la Capa de Transporte (TLS v1). Proporciona una API completa, cuya documentación se puede encontrar en *ssl(7)*.

8.49. Libelf de Elfutils-0.192

Libelf es una biblioteca para gestionar archivos ELF (formato ejecutable y enlazable).

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 135 MB

8.49.1. Instalación de Libelf

Libelf forma parte del paquete elfutils-0.192. Utilice el archivo elfutils-0.192.tar.bz2 como archivo tar de origen. Preparar Libelf para la compilación:

```
./configure --prefix=/usr         \
            --disable-debuginfod  \
            --enable-libdebuginfod=dummy
```

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar solo Libelf:

```
make -C libelf install
install -vm644 config/libelf.pc /usr/lib/pkgconfig
rm /usr/lib/libelf.a
```

8.49.2. Contenido de Libelf

Biblioteca instalada: libelf.so
Directorio de instalación: /usr/include/elfutils

Descripciones breves

`libelf.so` Contiene funciones de la API para gestionar archivos de objeto ELF

8.50. Libffi-3.4.7

La biblioteca Libffi proporciona una interfaz de programación portátil de alto nivel para diversas convenciones de llamada. Esto permite al programador llamar a cualquier función especificada por la descripción de la interfaz de llamada en tiempo de ejecución.

FFI significa Interfaz de Función Foránea (FFI). Una FFI permite que un programa escrito en un lenguaje llame a un programa escrito en otro. Específicamente, Libffi puede servir de puente entre un intérprete como Perl o Python y subrutinas de biblioteca compartida escritas en C o C++.

Tiempo de compilación aproximado: 1.7 SBU
Espacio en disco requerido: 11 MB

8.50.1. Instalación de Libffi

Nota

Al igual que GMP, Libffi se compila con optimizaciones específicas para el procesador en uso. Si se compila para otro sistema, cambie el valor del parámetro --with-gcc-arch= en el siguiente comando a un nombre de arquitectura completamente implementado por la CPU de ese sistema. Si no se hace esto, todas las aplicaciones que enlazan con libffi generarán errores de operación ilegal.

Prepare Libffi para la compilación:

```
./configure --prefix=/usr      \
            --disable-static   \
            --with-gcc-arch=native
```

Significado de la opción de configuración:

`--with-gcc-arch=native`

Asegúrese de que GCC se optimice para el sistema actual. Si no se especifica, se infiere el sistema y el código generado podría ser incorrecto. Si el código generado se copiará del sistema nativo a un sistema con menor capacidad, utilice este último como parámetro. Para obtener más información sobre los tipos de sistemas alternativos, consulte las opciones x86 en el manual de GCC.

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.50.2. Contenido de Libffi

Biblioteca instalada: libffi.so

Descripciones breves

`libffi`	Contiene las funciones API de la interfaz de funciones externas

8.51. Python-3.13.2

El paquete Python 3 contiene el entorno de desarrollo de Python. Es útil para la programación orientada a objetos, la escritura de scripts, la creación de prototipos de programas grandes y el desarrollo de aplicaciones completas. Python es un lenguaje de programación interpretado.

Tiempo de compilación aproximado: 2.1 SBU
Espacio en disco requerido: 501 MB

8.51.1. Instalación de Python 3

Preparar Python para la compilación:

```
./configure --prefix=/usr       \
            --enable-shared     \
            --with-system-expat \
            --enable-optimizations
```

Significado de las opciones de configuración:

`--with-system-expat`

Esta opción permite la vinculación con la versión del sistema de Expat.

`--enable-optimizations`

Esta opción permite pasos de optimización extensos, pero que requieren mucho tiempo. El intérprete se compila dos veces; las pruebas realizadas en la primera compilación se utilizan para mejorar la versión final optimizada. Compilar el paquete:

```
make
```

Se sabe que algunas pruebas se bloquean indefinidamente. Para comprobar los resultados, ejecute el conjunto de pruebas, pero establezca un límite de tiempo de 2 minutos para cada caso:

```
make test TESTOPTS="--timeout 120"
```

Para un sistema relativamente lento, podría ser necesario aumentar el límite de tiempo; 1 SBU (medido al compilar Binutils pass 1 con un núcleo de CPU) debería ser suficiente. Algunas pruebas son inestables, por lo que el conjunto de pruebas volverá a ejecutar automáticamente las pruebas fallidas. Si una prueba falló, pero luego pasó al volver a ejecutarse, debe considerarse como aprobada. Se sabe que una prueba, test_ssl, falla en el entorno chroot.

Instalar el paquete:

```
make install
```

En varios lugares de este libro, utilizamos el comando **pip3** para instalar programas y módulos de Python 3 para todos los usuarios como `root`. Esto contradice la recomendación de los desarrolladores de Python: instalar los paquetes en un entorno virtual o en el directorio personal de un usuario normal

(ejecutando **pip3** como este usuario). Se activa una advertencia de varias líneas cada vez que el usuario `root` ejecuta **pip3**.

La razón principal de esta recomendación es evitar conflictos con el gestor de paquetes del sistema (por ejemplo, **dpkg**).

LFS no cuenta con un gestor de paquetes para todo el sistema, por lo que esto no supone un problema. Además, pip3 buscará una nueva versión de sí mismo cada vez que se ejecute. Dado que la resolución de nombres de dominio aún no está configurada en el entorno chroot de LFS, **pip3** no puede buscar una nueva versión de sí mismo y generará una advertencia.

Tras iniciar el sistema LFS y configurar una conexión de red, se emitirá una advertencia diferente que indica al usuario que actualice **pip3** desde un wheel precompilado en PyPI (cuando haya una nueva versión disponible). Sin embargo, LFS considera pip3 como parte de Python 3, por lo que no debe actualizarse por separado. Además, una actualización desde un wheel precompilado se desviaría de nuestro objetivo: compilar un sistema Linux a partir del código fuente. Por lo tanto, la advertencia sobre una nueva versión de **pip3** también debe ignorarse. Si lo desea, puede suprimir todas estas advertencias ejecutando el siguiente comando, que crea un archivo de configuración:

```
cat > /etc/pip.conf << EOF
[global]
root-user-action = ignore
disable-pip-version-check = true
EOF
```

Importante

En LFS y BLFS, normalmente compilamos e instalamos módulos de Python con el comando pip3. Asegúrese de que los comandos pip3 install de ambos libros se ejecuten como usuario root (a menos que se trate de un entorno virtual de Python). Ejecutar pip3 install como usuario no root puede parecer eficaz, pero provocará que el módulo instalado sea inaccesible para otros usuarios.

pip3 install no reinstalará automáticamente un módulo ya instalado. Al usar el comando pip3 install para actualizar un módulo (por ejemplo, de meson-0.61.3 a meson-0.62.0), inserte la opción --upgrade
en la línea de comandos. Si es realmente necesario degradar un módulo o reinstalar la misma versión por algún motivo, inserte --force-reinstall --no-deps en la línea de comandos.

Si lo desea, instale la documentación preformateada:

```
install -v -dm755 /usr/share/doc/python-3.13.2/html

tar --strip-components=1                \
    --no-same-owner                     \
    --no-same-permissions               \
    -C /usr/share/doc/python-3.13.2/html \
    -xvf ../python-3.13.2-docs-html.tar.bz2
```

Significado de los comandos de instalación de la documentación:

`--no-same-owner y --no-same-permissions`

Asegúrese de que los archivos instalados tengan la propiedad y los permisos correctos. Sin estas opciones, tar instalará los archivos del paquete con los valores del creador original.

8.51.2. Contenido de Python 3

Programas instalados: 2to3, idle3, pip3, pydoc3, python3 y python3-config

Biblioteca instalada: libpython3.13.so y libpython3.so

Directorios instalados: /usr/include/python3.13, /usr/lib/python3 y /usr/share/doc/python-3.13.2

Descripciones breves

2to3 es un programa de Python que lee el código fuente de Python 2.x y aplica una serie de correcciones para transformarlo en código válido de Python 3.x.

idle3 es un script contenedor que abre un editor gráfico compatible con Python. Para que este script se ejecute, debe haber instalado Tk antes de Python para que se compile el módulo de Python Tkinter.

pip3 El instalador de paquetes para Python. Puedes usar pip para instalar paquetes desde el índice de paquetes de Python y otros índices.

pydoc3 es la herramienta de documentación de Python.

python3 es el intérprete de Python, un lenguaje de programación interpretado, interactivo y orientado a objetos.

8.52. Flit-Core-3.11.0

Flit-core es la parte de Flit (una herramienta de empaquetado para módulos Python simples) que permite compilar distribuciones.

Tiempo de compilación aproximado: Menos de 0.1 SBU
Espacio en disco requerido: 1.0 MB

8.52.1. Instalación de Flit-Core

Compilación del paquete:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instalación del paquete:

```
pip3 install --no-index --find-links dist flit_core
```

Significado de las opciones y comandos de configuración de pip3:

wheel

Este comando compila el archivo wheel para este paquete.

-w dist

Instruye a pip a colocar la rueda creada en el directorio dist.

--no-cache-dir

Evita que pip copie la rueda creada en el directorio /root/.cache/pip.

install

Este comando instala el paquete.

--no-build-isolation, --no-deps y --no-index

Estas opciones impiden la obtención de archivos del repositorio de paquetes en línea (PyPI). Si los paquetes se instalan en el orden correcto, pip no necesitará obtener ningún archivo; estas opciones ofrecen mayor seguridad en caso de error del usuario.

--find-links dist

Instruye a pip a buscar archivos wheel en el directorio dist.

8.52.2. Contenido de Flit-Core

Directorio de instalación: /usr/lib/python3.13/site-packages/flit_core y /usr/lib/python3.13/site-packages/flit_core-3.11.0.dist-info

8.53. Wheel-0.45.1

Wheel es una biblioteca de Python que constituye la implementación de referencia del estándar de empaquetado Wheel de Python.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 1,6 MB

8.53.1. Instalación de Wheel

Compile Wheel con el siguiente comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instale Wheel con el siguiente comando:

```
pip3 install --no-index --find-links dist wheel
```

8.53.2. Contenido de Wheel

Programa instalado: wheel

Directorios de instalación: /usr/lib/python3.13/site-packages/wheel y /usr/lib/python3.13/site-packages/wheel-0.45.1.dist-info

Descripciones breves

wheel es una utilidad para descomprimir, empaquetar o convertir archivos de wheel.

8.54. Setuptools-75.8.1

Setuptools es una herramienta que se utiliza para descargar, compilar, instalar, actualizar y desinstalar paquetes de Python.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 26 MB

8.54.1. Instalación de Setuptools

Compilación del paquete:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instalación del paquete:

```
pip3 install --no-index --find-links dist setuptools
```

8.54.2. Contenido de Setuptools

Directorio de instalación: /usr/lib/python3.13/site-packages/_distutils_hack, /usr/lib/python3.13/site-packages/pkg_resources, /usr/lib/python3.13/site-packages/setuptools y /usr/lib/python3.13/site-packages/setuptools-75.8.1.dist-info

8.55. Ninja-1.12.1

Ninja es un sistema de compilación pequeño enfocado en la velocidad.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 37 MB

8.55.1. Instalación de Ninja

Al ejecutarse, **ninja** normalmente utiliza el mayor número posible de procesos en paralelo. Por defecto, este es el número de núcleos del sistema, más dos. Esto puede sobrecalentar la CPU o provocar que el sistema se quede sin memoria. Al invocar **ninja** desde la línea de comandos, el parámetro -jN limitará el número de procesos en paralelo. Algunos paquetes integran la ejecución de **ninja** y no le pasan el parámetro -j.

El siguiente procedimiento opcional permite al usuario limitar el número de procesos en paralelo mediante una variable de entorno, NINJAJOBS. **Por ejemplo**, al establecer:

```
export NINJAJOBS=4
```

se limitará **ninja** a cuatro procesos en paralelo. Si lo desea, haga que **ninja** reconozca la variable de entorno NINJAJOBS ejecutando el editor de flujo:

```
sed -i '/int Guess/a \
  int   j = 0;\
  char* jobs = getenv( "NINJAJOBS" );\
  if ( jobs != NULL ) j = atoi( jobs );\
  if ( j > 0 ) return j;\
' src/ninja.cc
```

Construya Ninja con:

```
python3 configure.py --bootstrap --verbose
```

Significado de la opción de compilación:

`--bootstrap`

Este parámetro obliga a Ninja a reconstruirse para el sistema actual.

`--verbose`

Este parámetro hace que **configure.py** muestre el progreso de la compilación de Ninja.

Las pruebas del paquete no se pueden ejecutar en el entorno chroot. Requieren *`cmake`*. Sin embargo, la función básica de este paquete ya se ha probado reconstruyéndose (con la opción `--bootstrap`). Instalar el paquete:

```
install -vm755 ninja /usr/bin/
install -vDm644 misc/bash-completion /usr/share/bash-completion/completions/ninja
install -vDm644 misc/zsh-completion /usr/share/zsh/site-functions/_ninja
```

8.55.2. Contenido de Ninja

Programas instalados: ninja

Descripciones breves

ninja es el sistema de compilación de Ninja

8.56. Meson-1.7.0

Meson es un sistema de compilación de código abierto diseñado para ser extremadamente rápido y lo más intuitivo posible.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 44 MB

8.56.1. Instalación de Meson

Compile Meson con el siguiente comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

El conjunto de pruebas requiere algunos paquetes fuera del alcance de LFS. Instalar el paquete:

```
pip3 install --no-index --find-links dist meson
install -vDm644 data/shell-completions/bash/meson /usr/share/bash-completion/completions/meson
install -vDm644 data/shell-completions/zsh/_meson /usr/share/zsh/site-functions/_meson
```

Significado de los parámetros de instalación:

`-w dist`

Coloca las ruedas creadas en el directorio dist.

`--find-links dist`

Instala las ruedas desde el directorio dist.

8.56.2. Contenido de Meson

Programas instalados: meson

Directorio de instalación: /usr/lib/python3.13/site-packages/meson-1.7.0.dist-info y /usr/lib/python3.13/site-packages/mesonbuild

Descripciones breves

meson Un sistema de compilación de alta productividad

8.57. Kmod-34

El paquete Kmod contiene bibliotecas y utilidades para cargar módulos del kernel.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 11 MB

8.57.1. Instalación de Kmod

Preparar Kmod para la compilación:

```
mkdir -p build
cd       build

meson setup --prefix=/usr ..    \
            --sbindir=/usr/sbin \
            --buildtype=release \
            -D manpages=false
```

Significado de las opciones de configuración:

`-D manpages=false`

Esta opción deshabilita la generación de páginas de manual, lo que requiere un programa externo.

Compilar el paquete:

```
ninja
```

El conjunto de pruebas de este paquete requiere encabezados de kernel sin procesar (no los encabezados de kernel "saneados" instalados anteriormente), que están fuera del alcance de LFS.

Ahora instale el paquete:

```
ninja install
```

8.57.2. Contenido de Kmod

Programas instalados: depmod (enlace a kmod), insmod (enlace a kmod), kmod, lsmod (enlace a kmod), modinfo (enlace a kmod), modprobe (enlace a kmod) y rmmod (enlace a kmod)

Biblioteca instalada: libkmod.so

Descripciones breves

depmod Crea un archivo de dependencias basado en los símbolos que encuentra en el conjunto de módulos existente. Este archivo de dependencia lo utiliza **modprobe** para cargar automáticamente los módulos necesarios.

insmod Instala un módulo cargable en el kernel en ejecución.

kmod	Carga y descarga módulos del kernel.
lsmod	Enumera los módulos cargados actualmente.
modinfo	Examina un archivo de objeto asociado a un módulo del kernel y muestra cualquier nformación que pueda obtener.
modprobe	Utiliza un archivo de dependencia, creado por **depmod**, para cargar automáticamente los módulos relevantes.
rmmod	Descarga módulos del kernel en ejecución.
`libkmod`	Esta biblioteca la utilizan otros programas para cargar y descargar módulos del kernel.

8.58. Coreutils-9.6

El paquete Coreutils contiene las utilidades básicas necesarias para cualquier sistema operativo.

Tiempo de compilación aproximado: 1.2 SBU
Espacio en disco requerido: 182 MB

8.58.1. Instalación de Coreutils

POSIX requiere que los programas de Coreutils reconozcan correctamente los límites de caracteres, incluso en configuraciones regionales multibyte. El siguiente parche corrige este incumplimiento y otros errores relacionados con la internacionalización.

```
patch -Np1 -i ../coreutils-9.6-i18n-1.patch
```

Nota

Se han **encontrado** muchos errores en este parche. Al informar de nuevos errores a los responsables de Coreutils, compruebe primero si se pueden reproducir sin este parche.

Ahora prepare Coreutils para la compilación:

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

Significado de los comandos y opciones de configuración:

autoreconf -fv

El parche de internacionalización ha modificado el sistema de compilación, por lo que es necesario regenerar los archivos de configuración.

Normalmente, usaríamos la opción *-i* para actualizar los archivos auxiliares estándar, pero para este paquete no funciona porque `configure.ac` especificó una versión antigua de gettext.

automake -af

Autoreconf no actualizó los archivos auxiliares de automake debido a la falta de la opción *-i*. Este comando los actualiza para evitar un fallo de compilación.

`FORCE_UNSAFE_CONFIGURE=1`

Esta variable de entorno permite que el usuario root compile el paquete.

`--enable-no-install-program=kill,uptime`

El propósito de esta opción es evitar que Coreutils instale programas que serán instalados por o tros paquetes.

Compilar el paquete:

```
make
```

Si no se está ejecutando el conjunto de pruebas, vaya a "Instalar el paquete".

Ahora el conjunto de pruebas está listo para ejecutarse. Primero, ejecute las pruebas que deben ejecutarse como usuario `root`:

```
make NON_ROOT_USERNAME=tester check-root
```

Ejecutaremos el resto de las pruebas como el usuario tester. Algunas pruebas requieren que el usuario pertenezca a más de un grupo. Para evitar que estas pruebas se omitan, agregue un grupo temporal e incorpore al usuario `tester`:

```
groupadd -g 102 dummy -U tester
```

Corrija algunos de los permisos para que el usuario no root pueda compilar y ejecutar las pruebas:

```
chown -R tester .
```

Ahora ejecute las pruebas (usando `/dev/null` como entrada estándar; de lo contrario, dos pruebas podrían fallar si se compila LFS en una terminal gráfica o en una sesión en SSH o GNU Screen, ya que la entrada estándar está conectada a un PTY de la distribución del host y no se puede acceder al nodo del dispositivo para dicho PTY desde el entorno chroot de LFS):

```
su tester -c "PATH=$PATH make -k RUN_EXPENSIVE_TESTS=yes check" \
   < /dev/null
```

Elimine el grupo temporal:

```
groupdel dummy
```

Instale el paquete:

```
make install
```

Mueva los programas a las ubicaciones especificadas por el FHS:

```
mv -v /usr/bin/chroot /usr/sbin
mv -v /usr/share/man/man1/chroot.1 /usr/share/man/man8/chroot.8
sed -i 's/"1"/"8"/' /usr/share/man/man8/chroot.8
```

8.58.2. Contenido de Coreutils

Programas instalados: [, b2sum, base32, base64, basename, basenc, cat, chcon, chgrp, chmod, chown, chroot, cksum, comm, cp, csplit, cut, date, dd, df, dir, dircolors, dirname, du, echo, env, expand, expr, factor, false, fmt, fold, groups, head, hostid, id, install, join, link, ln, logname, ls, md5sum, mkdir, mkfifo, mknod, mktemp, mv, nice, nl, nohup, nproc, numfmt, od, paste, pathchk,

pinky, pr, printenv, printf, ptx, pwd, readlink, realpath, rm, rmdir, runcon, seq, sha1sum, sha224sum, sha256sum, sha384sum, sha512sum, shred, shuf, sleep, sort, split, stat, stdbuf, stty, sum, sync, tac, tail, tee, test, timeout, touch, tr, true, truncate, tsort, tty, uname, unexpand, uniq, unlink, users, vdir, wc, who, whoami, and yes

Biblioteca instalada: libstdbuf.so (en /usr/libexec/coreutils)

Directorios instalados: /usr/libexec/coreutils

Descripciones breves

[Es un comando real, /usr/bin/[; es sinónimo del comando de prueba.

Base32 Codifica y decodifica datos según la especificación base32 (RFC 4648).

base64 Codifica y decodifica datos según la especificación base64 (RFC 4648).

b2sum Imprime o comprueba sumas de comprobación de BLAKE2 (512 bits).

basename Elimina cualquier ruta y un sufijo dado de un nombre de archivo.

basenc Codifica o decodifica datos mediante varios algoritmos.

cat Concatena archivos a la salida estándar.

chcon Cambia el contexto de seguridad de archivos y directorios.

chgrp Cambia la propiedad del grupo de archivos y directorios.

chmod Cambia los permisos de cada archivo al modo especificado; el modo puede ser una representación simbólica de los cambios a realizar o un número octal que representa los nuevos permisos.

chown Cambia la propiedad del usuario o grupo de archivos y directorios.

chroot Ejecuta un comando con el directorio especificado como el directorio /.

cksum Imprime la suma de comprobación de redundancia cíclica (CRC) y el recuento de bytes de cada archivo especificado.

comm Compara dos archivos ordenados, mostrando en tres columnas las líneas únicas y comunes.

cp Copia archivos.

csplit Divide un archivo en varios archivos nuevos, separándolos según patrones o números de línea dados, y muestra el recuento de bytes de cada archivo.

cut Imprime secciones de líneas, seleccionando las partes según los campos o posiciones dados.

date Muestra la fecha y hora actuales en el formato dado o establece la fecha y hora del sistema.

dd	Copia un archivo utilizando el tamaño y el recuento de bloques dados, y opcionalmente realiza conversiones.
df	Informa la cantidad de espacio en disco disponible (y utilizado) en todos los sistemas de archivos montados, o solo en los sistemas de archivos que contienen los archivos seleccionados.
dir	Enumera el contenido de cada directorio dado (igual que el comando ls).
dircolors	Emite comandos para establecer LS_COLOR. Variable de entorno para cambiar el esquema de colores utilizado por ls.
dirname	Extrae las porciones de directorio del nombre especificado.
du	Informa la cantidad de espacio en disco utilizado por el directorio actual, por cada uno de los directorios especificados (incluidos todos los subdirectorios) o por cada uno de los archivos especificados.
echo	Muestra las cadenas especificadas.
env	Ejecuta un comando en un entorno modificado.
expand	Convierte tabulaciones en espacios.
expr	Evalúa expresiones.
factor	Imprime los factores primos de los enteros especificados.
false	No realiza ninguna acción, sin éxito. Siempre finaliza con un código de estado que indica un fallo.
fmt	Reformatea los párrafos en los archivos indicados.
fold	Ajusta las líneas en los archivos indicados.
groups	Informa sobre la pertenencia a grupos de un usuario.
head	Imprime las primeras diez líneas (o el número indicado) de cada archivo.
hostid	Informa sobre el identificador numérico (en hexadecimal) del host.
id	Informa sobre el ID de usuario efectivo, el ID de grupo y la pertenencia a grupos del usuario actual o del usuario especificado.
install	Copia archivos configurando sus modos de permisos y, si es posible, su propietario y grupo.
join	Une las líneas con campos de unión idénticos de dos archivos separados.
link	Crea un enlace físico (con el nombre indicado) a un archivo.
ln	Crea enlaces físicos o enlaces simbólicos entre archivos.
logname	Informa sobre el nombre de inicio de sesión del usuario actual.

ls	Enumera el contenido de cada directorio.
md5sum	Informa o comprueba las sumas de comprobación del Message Digest 5 (MD5).
mkdir	Crea directorios con los nombres dados.
mkfifo	Crea FIFO (First-In, First-Out), "tuberías con nombre" en el lenguaje UNIX, con los nombres dados.
mknod	Crea nodos de dispositivo con los nombres dados; un nodo de dispositivo es un archivo especial de caracteres, un archivo especial de bloques o un FIFO.
mktemp	Crea archivos temporales de forma segura; se utiliza en scripts.
mv	Mueve o renombra archivos o directorios.
nice	Ejecuta un programa con prioridad de programación modificada.
nl	Numera las líneas de los archivos dados.
nohup	Ejecuta un comando inmune a bloqueos, con su salida redirigida a un archivo de registro.
nproc	Imprime el número de unidades de procesamiento disponibles para un proceso.
numfmt	Convierte números a o desde cadenas legibles.
od	Voltea archivos en octal y otros formatos.
paste	Fusiona los archivos dados, uniendo secuencialmente las líneas correspondientes, una junto a la otra, separadas por tabulaciones.
pathchk	Comprueba si los nombres de archivo son válidos o portables.
pinky	Es un cliente ligero de finger. Reporta información sobre los usuarios.
pr	Pagina y organiza en columnas los archivos para su impresión.
printenv	Imprime el entorno.
printf	Imprime los argumentos según el formato indicado, similar a la función printf de C.
ptx	Genera un índice permutado a partir del contenido de los archivos, con cada palabra clave en su contexto.
pwd	Reporta el nombre del directorio de trabajo actual.
readlink	Reporta el valor del enlace simbólico.
realpath	Imprime la ruta resuelta.
rm	Elimina archivos o directorios.
rmdir	Elimina directorios si están vacíos.
runcon	Ejecuta un comando con el contexto de seguridad especificado.

seq	Imprime una secuencia de números dentro de un rango y con un incremento dados.
sha1sum	Imprime o verifica sumas de comprobación del algoritmo hash seguro 1 (SHA1) de 160 bits.
sha224sum	Imprime o verifica sumas de comprobación del algoritmo hash seguro de 224 bits.
sha256sum	Imprime o verifica sumas de comprobación del algoritmo hash seguro de 256 bits.
sha384sum	Imprime o verifica sumas de comprobación del algoritmo hash seguro de 384 bits.
sha512sum	Imprime o verifica sumas de comprobación del algoritmo hash seguro de 512 bits. Sumas de comprobación de algoritmos
shred	Sobrescribe repetidamente los archivos con patrones complejos, lo que dificulta la recuperación de datos
shuf	Reordena las líneas de texto
sleep	Hace una pausa durante el tiempo especificado
sort	Ordena las líneas de los archivos
split	Divide el archivo en fragmentos, por tamaño o por número de líneas
stat	Muestra el estado del archivo o del sistema de archivos.
stdbuf	Ejecuta comandos con operaciones de almacenamiento en búfer modificadas para sus flujos estándar.
stty	Establece o informa la configuración de la línea terminal.
sum	Imprime la suma de comprobación y el recuento de bloques de cada archivo.
sync	Limpia los búferes del sistema de archivos; fuerza la transferencia de los bloques modificados al disco y actualiza el superbloque.
tac	Concatena los archivos en orden inverso.
tail	Imprime las últimas diez líneas (o el número especificado) de cada archivo.
tee	Lee desde la entrada estándar mientras escribe tanto en la salida estándar como en los archivos.
test	Compara valores y comprueba los tipos de archivo.
timeout	Ejecuta un comando con un límite de tiempo.
touch	Cambia las marcas de tiempo de los archivos, estableciendo las horas de acceso y modificación de los archivos a la hora actual; Los archivos inexistentes se crean con longitud cero.
tr	Traduce, comprime y elimina los caracteres de la entrada estándar.

true	No realiza ninguna acción correctamente; siempre termina con un código de estado que indica éxito.
truncate	Comprime o expande un archivo al tamaño especificado.
tsort	Realiza una ordenación topológica. Escribe una lista completamente ordenada según el orden parcial de un archivo dado.
tty	Informa del nombre de archivo del terminal conectado a la entrada estándar.
uname	Informa de la información del sistema.
unexpand	Convierte espacios en tabulaciones.
uniq	Descarta todas las líneas idénticas sucesivas excepto una.
unlink	Elimina el archivo dado.
users	Informa de los nombres de los usuarios conectados.
vdir	Es lo mismo que `ls -l`.
wc	Informa del número de líneas, palabras y bytes de cada archivo dado, así como los totales generales cuando se proporciona más de un archivo.
who	Informa de quién ha iniciado sesión.
whoami	Informa del nombre de usuario asociado con el ID de usuario efectivo actual.
yes	Emite repetidamente `y` o una cadena dada, hasta que se elimina.
`libstdbuf`	Biblioteca utilizada por `stdbuf`.

8.59. Check-0.15.2

Check es un framework de pruebas unitarias para C.

Tiempo de compilación aproximado: 0.1 SBU (aproximadamente 2.1 SBU con pruebas)
Espacio en disco requerido: 11 MB

8.59.1. Instalación de Check

Preparar Check para la compilación:

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

Compilación del paquete:

```
make
```

La compilación ha finalizado. Para ejecutar el conjunto de pruebas de Check, ejecute el siguiente comando:

```
make check
```

Instalar el paquete:

```
make docdir=/usr/share/doc/check-0.15.2 install
```

8.59.2. Contenido de Check

Programa instalado: checkmk
Biblioteca instalada: libcheck.so

Descripciones breves

checkmk Script de AWK para generar pruebas unitarias de C para usar con el framework de pruebas unitarias Check

`libcheck.so` Contiene funciones que permiten llamar a Check desde un programa de prueba

8.60. Diffutils-3.11

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

Tiempo de compilación aproximado: 0.4 SBU
Espacio en disco requerido: 50 MB

8.60.1. Instalación de Diffutils

Preparar Diffutils para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.60.2. Contenido de Diffutils

Programas instalados: cmp, diff, diff3 y sdiff

Descripciones breves

cmp Compara dos archivos e informa de las diferencias byte a byte

diff Compara dos archivos o directorios e informa de las líneas que difieren

diff3 Compara tres archivos línea a línea

sdiff Combina dos archivos y muestra los resultados de forma interactiva

8.61. Gawk-5.3.1

El paquete Gawk contiene programas para manipular archivos de texto.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 43 MB

8.61.1. Instalación de Gawk

Primero, asegúrese de que no se instalen archivos innecesarios:

```
sed -i 's/extras//' Makefile.in
```

Prepare Gawk para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Instalar el paquete:

```
rm -f /usr/bin/gawk-5.3.1
make install
```

Significado del comando:

rm -f /usr/bin/gawk-5.3.1

El sistema de compilación no recreará el enlace físico gawk-5.3.1 si ya existe. Elimínelo para garantizar que el enlace físico anterior instalado en la Sección 6.9, “Gawk-5.3.1” se actualice aquí.

El proceso de instalación ya creó **awk** como enlace simbólico a **gawk**; cree también su página de manual como enlace simbólico:

```
ln -sv gawk.1 /usr/share/man/man1/awk.1
```

Si lo desea, instale la documentación:

```
install -vDm644 doc/{awkforai.txt,*.{eps,pdf,jpg}} -t /usr/share/doc/gawk-5.3.1
```

8.61.2. Contenido de Gawk

Programas instalados: awk (enlace a gawk), gawk y gawk-5.3.1

Bibliotecas instaladas: filefuncs.so, fnmatch.so, fork.so, inplace.so, intdiv.so, ordchr.so, readdir.so, readfile.so, revoutput.so, revtwoway.so, rwarray.so y time.so (todas en /usr/lib/gawk)

Directorios instalados: /usr/lib/gawk, /usr/libexec/awk, /usr/share/awk y /usr/share/doc/gawk-5.3.1

Descripciones breves

awk Enlace a **gawk**

gawk Un programa para manipular archivos de texto; es la implementación GNU de **awk**

gawk-5.3.1 Enlace físico a **gawk**

8.62. Findutils-4.10.0

El paquete Findutils contiene programas para buscar archivos. Se proporcionan programas para buscar en todos los archivos de un árbol de directorios y para crear, mantener y buscar en una base de datos (a menudo más rápido que la búsqueda recursiva, pero poco fiable a menos que la base de datos se haya actualizado recientemente). Findutils también incluye el programa **xargs**, que permite ejecutar un comando específico en cada archivo seleccionado en una búsqueda.

Tiempo de compilación aproximado: 0.7 SBU
Espacio en disco requerido: 63 MB

8.62.1. Instalación de Findutils

Preparar Findutils para la compilación:

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

Significado de las opciones de configuración:

`--localstatedir`

Esta opción mueve la base de datos de localización a `/var/lib/locate`, que es la ubicación compatible con FHS.

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Instalar el paquete:

```
make install
```

8.62.2. Contenido de Findutils

Programas instalados: find, locate, updatedb y xargs
Directorio de instalación: /var/lib/locate

Descripciones breves

find Busca archivos que coincidan con los criterios especificados en los árboles de directorios

locate Busca en una base de datos de nombres de archivo e informa de los nombres que contienen una cadena o coinciden con un patrón

updatedb Actualiza la base de datos **locate**; escanea todo el sistema de archivos (incluidos otros sistemas de archivos montados, a menos que se indique lo contrario) y guarda todos los nombres de archivo que encuentra en la base de datos

xargs Se puede usar para aplicar un comando a una lista de archivos

8.63. Groff-1.23.0

El paquete Groff contiene programas para procesar y formatear texto e imágenes.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 108 MB

8.63.1. Instalación de Groff

Groff espera que la variable de entorno PAGE contenga el tamaño de papel predeterminado. Para usuarios en Estados Unidos, PAGE=letter es adecuado. En otros lugares, PAGE=A4 puede ser más adecuado. Aunque el tamaño de papel predeterminado se configura durante la compilación, se puede sobrescribir posteriormente indicando "A4" o "letter" en el archivo /etc/papersize.

Preparar Groff para la compilación:

```
PAGE=<paper_size> ./configure --prefix=/usr
```

Compilación del paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.63.2. Contenido de Groff

Programas instalados: addftinfo, afmtodit, chem, eqn, eqn2graph, gdiffmk, glilypond, gperl, gpinyin, grap2graph, grn, grodvi, groff, groffer, grog, grolbp, grolj4, gropdf, grops, grotty, hpftodit, indxbib, lkbib, lookbib, mmroff, neqn, nroff, pdfmom, pdfroff, pfbtops, pic, pic2graph, post-grohtml, preconv, pre-grohtml, refer, roff2dvi, roff2html, roff2pdf, roff2ps, roff2text, roff2x, soelim, tbl, tfmtodit y troff

Directorios instalados: /usr/lib/groff y /usr/share/doc/groff-1.23.0, /usr/share/groff

Descripciones breves

addftinfo Lee un archivo de fuente troff y añade información adicional sobre la métrica de la fuente, utilizada por el sistema **groff**.

afmtodit Crea un archivo de fuente para usar con **groff** y **grops**.

chem Preprocesador de Groff para generar diagramas de estructura química.

eqn Compila descripciones de ecuaciones integradas en archivos de entrada de troff en comandos que **troff** entiende.

eqn2graph Convierte una ecuación de troff en una imagen recortada.

gdiffmk Marca las diferencias entre archivos groff/nroff/troff.

glilypond Transforma partituras escritas en el lenguaje lilypond al lenguaje groff.

gperl Preprocesador para groff, que permite la inserción de código Perl en archivos groff.

gpinyin Preprocesador para groff, que permite la inserción de Pinyin (chino mandarín escrito con el alfabeto romano) en archivos groff.

grap2graph Convierte un archivo de programa grap en una imagen de mapa de bits recortada (grap es un antiguo lenguaje de programación Unix para crear diagramas).

grn Un preprocesador de **groff** para archivos gremlin.

grodvi Un controlador para **groff** que produce archivos de salida en formato TeX dvi.

groff Una interfaz para el sistema de formato de documentos groff. Normalmente, ejecuta el programa **troff** y un postprocesador adecuado para el dispositivo seleccionado.

groffer Muestra archivos y páginas de manual de **groff** en terminales X y tty.

grog Lee archivos y determina cuáles de las opciones de **groff** -e, -man, -me, -mm, -ms, -p, -s y -t son necesarias para imprimir archivos, e informa del comando **groff**, incluyendo dichas opciones.

grolbp Es un controlador de **groff** para impresoras Canon CAPSL (impresoras láser de las series LBP-4 y LBP-8).

grolj4 Es un controlador para **groff** que produce resultados en formato PCL5, compatible con impresoras HP LaserJet 4.

gropdf Traduce los resultados de GNU **troff** a PDF.

grops Traduce los resultados de GNU **troff** a PostScript.

grotty Traduce los resultados de GNU **troff** a un formato compatible con dispositivos tipo máquina de escribir.

hpftodit Crea un archivo de fuentes para usar con **groff -Tlj4** a partir de un archivo de métricas de fuentes con etiquetas HP.

indxbib Crea un índice invertido para las bases de datos bibliográficas con un archivo específico para usar con **refer**, **lookbib** y **lkbib**

lkbib Busca en bases de datos bibliográficas referencias que contengan claves específicas e informa de las referencias encontradas.

lookbib Imprime un mensaje en el error estándar (a menos que la entrada estándar no sea una terminal), lee una línea con un conjunto de palabras clave de la entrada estándar, busca en las bases de datos bibliográficas de un archivo especificado referencias que contengan esas palabras clave, imprime las referencias encontradas en la salida estándar y repite este proceso hasta el final de la entrada.

mmroff	Un preprocesador simple para **groff**
neqn	Formatea ecuaciones para la salida ASCII (Código Estándar Americano para el Intercambio de Información).
nroff	Un script que emula el comando **nroff** usando **groff**.
pdfmom	Es un contenedor de groff que facilita la producción de documentos PDF a partir de archivos formateados con las macros mom.
pdfroff	Crea documentos PDF con groff.
pfbtops	Traduce una fuente PostScript en formato .pfb a ASCII.
pic	Compila descripciones de imágenes incrustadas en archivos de entrada de troff o TeX en comandos que TeX o **troff** entiende.
pic2graph	Convierte un diagrama PIC en una imagen recortada.
post-grohtml	Traduce la salida de GNU **troff** a HTML.
preconv	Convierte la codificación de los archivos de entrada a un formato que GNU **troff** entiende.
pre-grohtml	Traduce la salida de GNU **troff** a HTML.
refer	Copia el contenido de un archivo a la salida estándar, excepto que las líneas entre .[y .] se interpretan como citas, y las líneas entre .R1 y .R2 se interpretan como comandos que indican cómo deben procesarse las citas.
roff2dvi	Transforma archivos roff al formato DVI
roff2html	Transforma archivos roff al formato HTML
roff2pdf	Transforma archivos roff a PDF
roff2ps	Transforma archivos roff a archivos ps
roff2text	Transforma archivos roff a archivos de texto
roff2x	Transforma archivos roff a otros formatos
soelim	Lee archivos y reemplaza líneas del formato .so por el contenido del archivo mencionado
tbl	Compila descripciones de tablas incrustadas en archivos de entrada troff en comandos que **troff** entiende
tfmtodit	Crea un archivo de fuentes para usar con **groff -Tdvi**
troff	Es altamente compatible con **troff** de Unix; normalmente se debe invocar con el comando **groff**, que también ejecutará preprocesadores y posprocesadores en el orden y con las opciones adecuadas

8.64. GRUB-2.12

El paquete GRUB contiene el cargador de arranque unificado GRand.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 166 MB

8.64.1. Instalación de GRUB

Nota

Si su sistema es compatible con UEFI y desea arrancar LFS con UEFI, debe instalar GRUB con soporte UEFI (y sus dependencias) siguiendo las instrucciones de la *página de BLFS*. Puede omitir este paquete o instalarlo junto con el paquete GRUB para UEFI de BLFS sin conflicto (la página de BLFS proporciona instrucciones para ambos casos).

Advertencia

Desactive cualquier variable de entorno que pueda afectar la compilación:

```
unset {C,CPP,CXX,LD}FLAGS
```

No intente ajustar este paquete con indicadores de compilación personalizados. Este paquete es un cargador de arranque. Las operaciones de bajo nivel del código fuente podrían verse afectadas por una optimización excesiva.

Agregar un archivo que falta en el archivo tarball de la versión:

```
echo depends bli part_gpt > grub-core/extra_deps.lst
```

Preparar GRUB para la compilación:

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

Significado de las nuevas opciones de configuración:

`--disable-werror`

Esto permite que la compilación se complete con las advertencias introducidas por las versiones más recientes de Flex.

`--disable-efiemu`

Esta opción minimiza la compilación al deshabilitar una función y eliminar algunos programas de prueba innecesarios para LFS.

Compilar el paquete:

```
make
```

No se recomienda el conjunto de pruebas para estos paquetes. La mayoría de las pruebas dependen de paquetes que no están disponibles en el entorno LFS limitado. Para ejecutar las pruebas de todos modos, ejecute **make check**.

Instale el paquete y mueva el archivo de soporte de compleción de Bash a la ubicación recomendada por los responsables de la compleción de Bash:

```
make install
mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

Cómo hacer que su sistema LFS sea arrancable con GRUB se explicará en la Sección 10.4, “Uso de GRUB para configurar el proceso de arranque”.

8.64.2. Contenido de GRUB

Programas instalados: grub-bios-setup, grub-editenv, grub-file, grub-fstest, grub-glue-efi, grub-install, grub-kbdcomp, grub-macbless, grub-menulst2cfg, grub-mkconfig, grub-mkimage, grub-mklayout, grub-mknetdir, grub-mkpasswd-pbkdf2, grub-mkrelpath, grub-mkrescue, grub-mkstandalone, grub-ofpathname, grub-probe, grub-reboot, grub-render-label, grub-script-check, grub-set-default, grub-sparc64-setup, and grub-syslinux2cfg

Directorios instalados: /usr/lib/grub, /etc/grub.d, /usr/share/grub, and /boot/grub (when grub-install is first run)

Descripciones breves

grub-bios-setup Es un programa auxiliar para **grub-install**.

grub-editenv Es una herramienta para editar el bloque de entorno.

grub-file Comprueba si el archivo dado es del tipo especificado.

grub-fstest Es una herramienta para depurar el controlador del sistema de archivos.

grub-glue-efi Une binarios de 32 y 64 bits en un solo archivo (para equipos Apple).

grub-install Instala GRUB en la unidad.

grub-kbdcomp Es un script que convierte una distribución xkb en una reconocida por GRUB.

grub-macbless Es el bless de estilo Mac para sistemas de archivos HFS o HFS+ (**bless** es exclusivo de los equipos Apple; permite que un dispositivo sea arrancable).

grub-menulst2cfg Convierte un GRUB Legacy menu.lst en un grub.cfg para su uso con GRUB 2.

grub-mkconfig Genera un archivo grub.cfg.

grub-mkimage Crea una imagen de arranque de GRUB.

grub-mklayout Genera un archivo de distribución de teclado de GRUB.

grub-mknetdir Prepara un directorio de arranque de red de GRUB.

grub-mkpasswd-pbkdf2	Genera una contraseña PBKDF2 cifrada para su uso en el menú de arranque.
grub-mkrelpath	Convierte una ruta del sistema en una ruta relativa a su directorio raíz.
grub-mkrescue	Crea una imagen de arranque de GRUB adecuada para un disquete, CDROM/DVD o una unidad USB.
grub-mkstandalone	Genera una imagen independiente.
grub-ofpathname	Es un programa auxiliar que imprime la ruta de un dispositivo GRUB.
grub-probe	Sondea la información del dispositivo para una ruta o dispositivo determinado.
grub-reboot	Establece la entrada de arranque predeterminada de GRUB solo para el siguiente arranque.
grub-render-label	Representa el archivo .disk_label de Apple para Macs.
grub-script-check	Comprueba el script de configuración de GRUB en busca de errores de sintaxis.
grub-set-default	Establece la entrada de arranque predeterminada de GRUB.
grub-sparc64-setup	Es un programa auxiliar para grub-setup.
grub-syslinux2cfg	Transforma un archivo de configuración de syslinux al formato grub.cfg.

8.65. Gzip-1.13

El paquete Gzip contiene programas para comprimir y descomprimir archivos.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 21 MB

8.65.1. Instalación de Gzip

Preparar Gzip para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.65.2. Contenido de Gzip

Programas instalados: gunzip, gzexe, gzip, uncompress (enlace duro con gunzip), zcat, zcmp, zdiff, zegrep, zfgrep, zforce, zgrep, zless, zmore y znew

Descripciones breves

gunzip Descomprime archivos comprimidos

gzexe Crea archivos ejecutables autodescomprimibles

gzip Comprime los archivos proporcionados mediante codificación Lempel-Ziv (LZ77)

uncompress Descomprime archivos comprimidos

zcat Descomprime los archivos comprimidos a la salida estándar

zcmp Ejecuta **cmp** en archivos comprimidos

zdiff Ejecuta **diff** en archivos comprimidos

zegrep Ejecuta **egrep** en archivos comprimidos

zfgrep Ejecuta **fgrep** en archivos comprimidos

zforce	Fuerza la extensión `.gz` en todos los archivos comprimidos con **gzip**, para que gzip no los vuelva a comprimir; esto puede ser útil cuando los nombres de archivo se truncaron durante una transferencia.
zgrep	Ejecuta **grep** en archivos comprimidos con gzip.
zless	Se ejecuta **less** en archivos comprimidos con gzip.
zmore	Se ejecuta **more** en archivos comprimidos con gzip.
znew	**Recomprime** archivos del formato comprimido al formato **gzip** (de .z a .gz).

8.66. IPRoute2-6.13.0

El paquete IPRoute2 contiene programas para redes básicas y avanzadas basadas en IPV4.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 17 MB

8.66.1. Instalación de IPRoute2

El programa arpd incluido en este paquete no se compilará, ya que depende de la base de datos Berkeley, que no está instalada en LFS. Sin embargo, se instalarán un directorio y una página de manual para arpd. Para evitarlo, ejecute los comandos que se muestran a continuación:

```
sed -i /ARPD/d Makefile
rm -fv man/man8/arpd.8
```

Compilar el paquete:

```
make NETNS_RUN_DIR=/run/netns
```

Este paquete no cuenta con un conjunto de pruebas funcional. Instalar el paquete:

```
make SBINDIR=/usr/sbin install
```

Si lo desea, instale la documentación:

```
install -vDm644 COPYING README* -t /usr/share/doc/iproute2-6.13.0
```

8.66.2. Contenido de IPRoute2

Programas instalados: bridge, ctstat (enlace a lnstat), genl, ifstat, ip, lnstat, nstat, routel, rtacct, rtmon, rtpr, rtstat (enlace a lnstat), ss y tc

Directorios instalados: /etc/iproute2, /usr/lib/tc y /usr/share/doc/iproute2-6.13.0

Descripciones breves

bridge Configura puentes de red

ctstat Utilidad de estado de conexión

genl Interfaz genérica de la utilidad netlink

ifstat Muestra estadísticas de interfaz, incluyendo el número de paquetes transmitidos y recibidos, por interfaz

ip El ejecutable principal. Tiene varias funciones, incluyendo las siguientes:

ip link <device> permite a los usuarios ver el estado de los dispositivos y realizar cambios

i**p addr** permite a los usuarios ver direcciones y sus propiedades, añadir nuevas direcciones y eliminar las antiguas

ip neighbor permite a los usuarios ver enlaces de vecinos y sus propiedades, añadir nuevas entradas de vecinos y eliminar las antiguas

ip rule permite a los usuarios ver las políticas de enrutamiento y modificarlas

ip route permite a los usuarios ver la tabla de enrutamiento y modificar sus reglas

ip tunnel permite a los usuarios ver los túneles IP y sus propiedades y modificarlas

ip maddr permite a los usuarios ver las direcciones multicast y sus propiedades y modificarlas

ip mroute permite a los usuarios configurar, cambiar o eliminar el enrutamiento multicast

ip monitor permite a los usuarios supervisar continuamente el estado de dispositivos, direcciones y rutas

lnstat Proporciona estadísticas de red de Linux; es un reemplazo generalizado y con más funciones del antiguo programa **rtstat**.

nstat Muestra estadísticas de red.

routel Un componente de **ip route** para listar las tablas de enrutamiento.

rtacct Muestra el contenido de `/proc/net/rt_acct`.

rtmon Utilidad de monitorización de rutas.

rtpr Convierte la salida de **ip -o** a un formato legible.

rtstat Utilidad de estado de ruta.

ss Similar al comando **netstat**; muestra las conexiones activas.

tc Control de tráfico para implementaciones de calidad de servicio (QoS) y clase de servicio (CoS).

tc qdisc permite a los usuarios configurar la disciplina de colas.

tc class permite a los usuarios configurar clases según la programación de la disciplina de colas.

tc filter permite a los usuarios configurar el filtrado de paquetes QoS/CoS.

tc monitor permite visualizar los cambios realizados en el control de tráfico del kernel.

8.67. Kbd-2.7.1

El paquete Kbd contiene archivos de tabla de teclas, fuentes de consola y utilidades de teclado.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 34 MB

8.67.1. Instalación de Kbd

El comportamiento de las teclas de retroceso y suprimir no es uniforme en los mapas de teclas del paquete Kbd. El siguiente parche corrige este problema para los mapas de teclas i386:

```
patch -Np1 -i ../kbd-2.7.1-backspace-1.patch
```

Tras la aplicación del parche, la tecla de retroceso genera el carácter con el código 127, y la tecla suprimir genera una secuencia de escape conocida.
Elimine el programa redundante **resizecons** (requiere la extinta `svgalib` para proporcionar los archivos de modo de vídeo; para un uso normal, `**setfont**` ajusta el tamaño de la consola correctamente) junto con su página de manual.

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

Preparar Kbd para la compilación:

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

Significado de la opción configure:

`--disable-vlock`
: Esta opción impide la compilación de la utilidad vlock porque requiere la biblioteca PAM, que no está disponible en el entorno chroot.

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Nota

Para algunos idiomas (p. ej., bielorruso), el paquete Kbd no proporciona una configuración de teclado útil, ya que la configuración estándar "by" asume la codificación ISO-8859-5 y normalmente se utiliza la configuración CP1251. Los usuarios de estos idiomas deben descargar las configuraciones de teclado funcionales por separado.

Si lo desea, instale la documentación:

```
cp -R -v docs/doc -T /usr/share/doc/kbd-2.7.1
```

8.67.2. Contenido de Kbd

Programas instalados: chvt, deallocvt, dumpkeys, fgconsole, getkeycodes, kbdinfo, kbd_mode, kbdrate, loadkeys, loadunimap, mapscrn, openvt, psfaddtable (enlace a psfxtable), psfgettable (enlace a psfxtable), psfstriptable (enlace a psfxtable), psfxtable, setfont, setkeycodes, setleds, setmetamode, setvtrgb, showconsolefont, showkey, unicode_start y unicode_stop

Directorios instalados: /usr/share/consolefonts, /usr/share/consoletrans, /usr/share/keymaps, /usr/share/doc/kbd-2.7.1 y /usr/share/unimaps

Descripciones breves

chvt Cambia la terminal virtual en primer plano.

deallocvt Desasigna las terminales virtuales no utilizadas.

dumpkeys Voltea las tablas de traducción del teclado.

fgconsole Imprime el número de la terminal virtual activa.

getkeycodes Imprime la tabla de mapeo de código de escaneo a código de tecla del kernel.

kbdinfo Obtiene información sobre el estado de una consola.

kbd_mode Informa o establece el modo del teclado.

kbdrate Establece las tasas de repetición y retardo del teclado.

loadkeys Carga las tablas de traducción del teclado.

loadunimap Carga la tabla de mapeo de Unicode a fuente del kernel.

mapscrn Un programa obsoleto solía cargar una tabla de mapeo de caracteres de salida definida por el usuario en el controlador de la consola; esto ahora lo realiza **setfont**.

openvt Inicia un programa en una nueva terminal virtual (VT).

psfaddtable Agrega una tabla de caracteres Unicode a una fuente de consola.

psfgettable Extrae la tabla de caracteres Unicode incrustada de una fuente de consola.

psfstriptable Elimina la tabla de caracteres Unicode incrustada de una fuente de consola.

psfxtable Administra las tablas de caracteres Unicode para fuentes de consola.

setfont Modifica las fuentes Enhanced Graphic Adapter (EGA) y Video Graphics Array (VGA) en la consola.

setkeycodes Carga las entradas de la tabla de mapeo de código de escaneo a código de tecla del kernel. Esto es útil si hay teclas inusuales en el teclado.

setleds Establece las banderas del teclado y los diodos emisores de luz (LED).

setmetamode Define el manejo de metateclas del teclado.

setvtrgb Establece el mapa de colores de la consola en todas las terminales virtuales.

showconsolefont Muestra la fuente actual de la pantalla de la consola EGA/VGA.

showkey Informa los códigos de escaneo, los códigos de tecla y los códigos ASCII de las teclas pulsadas.

unicode_start Pone el teclado y la consola en modo UNICODE. [No utilice este programa a menos que su archivo de mapa de teclas tenga la codificación ISO-8859-1. Para otras codificaciones, esta utilidad produce resultados incorrectos].

unicode_stop Revierte el teclado y la consola del modo UNICODE.

8.68. Libpipeline-1.5.8

El paquete Libpipeline contiene una biblioteca para manipular pipelines de subprocesos de forma flexible y práctica.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 11 MB

8.68.1. Instalación de Libpipeline

Preparar Libpipeline para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.68.2. Contenido de Libpipeline

Biblioteca instalada: libpipeline.so

Descripciones breves

`libpipeline` Esta biblioteca se utiliza para construir pipelines entre subprocesos de forma segura

8.69. Make-4.4.1

El paquete Make contiene un programa para controlar la generación de ejecutables y otros archivos no fuente de un paquete a partir de los archivos fuente.

Tiempo de compilación aproximado: 0.7 SBU
Espacio en disco requerido: 13 MB

8.69.1. Instalación de Make

Preparar Make para la compilación:

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

Compilar el paquete:

```
make
```

Para probar los resultados, ejecute:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Instalar el paquete:

```
make install
```

8.69.2. Contenido de Make

Programa instalado: make

Descripciones breves

make Determina automáticamente qué partes de un paquete deben (re)compilarse y luego ejecuta los comandos correspondientes.

8.70. Parche-2.7.6

El paquete de parches contiene un programa para modificar o crear archivos mediante la aplicación de un archivo de parche, generalmente creado por el programa **diff**.

Tiempo de compilación aproximado: 0.2 SBU
Espacio en disco requerido: 12 MB

8.70.1. Instalación del parche

Preparar el parche para la compilación:

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

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.70.2. Contenido del parche

Programa instalado: patch

Descripciones breves

patch Modifica los archivos según un archivo de parche (un archivo de parche suele ser una lista de diferencias creada con el programa **diff**. Al aplicar estas diferencias a los archivos originales, **patch** crea las versiones parcheadas).

8.71. Tar-1.35

El paquete Tar permite crear archivos tar, así como realizar otras manipulaciones de archivos. Tar se puede usar en archivos previamente creados para extraer archivos, almacenar archivos adicionales, actualizar o listar archivos ya almacenados.

Tiempo de compilación aproximado: 0.6 SBU
Espacio en disco requerido: 43 MB

8.71.1. Instalación de Tar

Preparar Tar para la compilación:

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

Significado de la opción de configuración:

FORCE_UNSAFE_CONFIGURE=1

Esto obliga a que la prueba de `mknod` se ejecute como `root`. Generalmente se considera peligroso ejecutar esta prueba como `root`, pero como se ejecuta en un sistema que solo se ha compilado parcialmente, no hay problema en sobrescribirla.

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Se sabe que una prueba, "capacidades: almacenamiento/restauración de binarios", falla si se ejecuta porque LFS no es compatible con SELinux, pero se omitirá si el kernel del host no admite atributos extendidos ni etiquetas de seguridad en el sistema de archivos utilizado para compilar LFS.

Instalar el paquete:

```
make install
make -C doc install-html docdir=/usr/share/doc/tar-1.35
```

8.71.2. Contenido de Tar

Programas instalados: tar

Directorio de instalación: /usr/share/doc/tar-1.35

Descripciones breves

tar Crea, extrae archivos y lista su contenido en archivos comprimidos, también conocidos como tarballs.

8.72. Texinfo-7.2

El paquete Texinfo contiene programas para leer, escribir y convertir páginas de información.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 160 MB

8.72.1. Instalación de Texinfo

Preparar Texinfo para la compilación:

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

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

Opcionalmente, instale los componentes de una instalación de TeX:

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

Significado del parámetro make:

TEXMF=/usr/share/texmf

La variable makefile de TEXMF contiene la ubicación de la raíz del árbol de TeX si, por ejemplo, se instalará un paquete de TeX posteriormente.

El sistema de documentación de Info utiliza un archivo de texto plano para guardar su lista de entradas de menú. El archivo se encuentra en /usr/share/info/dir. Desafortunadamente, debido a problemas ocasionales con los Makefiles de varios paquetes, a veces puede desincronizarse con las páginas de información instaladas en el sistema. Si alguna vez es necesario recrear el archivo /usr/share/info/dir, los siguientes comandos opcionales lo harán:

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

8.72.2. Contenido de Texinfo

Programas instalados: info, install-info, makeinfo (enlace a texi2any), pdftexi2dvi, pod2texi, texi2any, texi2dvi, texi2pdf y texindex

Bibliotecas instaladas: MiscXS.so, Parsetexi.so y XSParagraph.so (todas en /usr/lib/texinfo)

Directorios instalados: /usr/share/texinfo y /usr/lib/texinfo

Descripciones breves

info Se utiliza para leer páginas de información similares a las páginas de manual, pero que a menudo profundizan mucho más que simplemente explicar todas las opciones disponibles de la línea de comandos [Por ejemplo, compare **man bison** e **info bison**].

install-info Se utiliza para instalar páginas de información; actualiza las entradas en el archivo de índice de **información**.

makeinfo Traduce los documentos fuente de Texinfo a páginas de información, texto plano o HTML.

pdftexi2dvi Se utiliza para formatear el documento de Texinfo a un archivo PDF (Portable Document Format).

pod2texi Convierte Pod a formato Texinfo

texi2any Traduce la documentación fuente de Texinfo a otros formatos.

texi2dvi Se utiliza para formatear el documento Texinfo en un archivo independiente del dispositivo que se pueda imprimir.

texi2pdf Se utiliza para formatear el documento Texinfo en un archivo PDF (Portable Document Format).

texindex Se utiliza para ordenar los archivos de índice de Texinfo.

8.73. Vim-9.1.1166

El paquete Vim contiene un potente editor de texto.

Tiempo de compilación aproximado: 3.4 SBU
Espacio en disco requerido: 251 MB

Alternativas a Vim

Si prefiere otro editor, como Emacs, Joe o Nano, consulte

https://www.linuxfromscratch.org/blfs/view/12.3/postlfs/editors.html

para obtener instrucciones de instalación sugeridas.

8.73.1. Instalación de Vim

Primero, cambie la ubicación predeterminada del archivo de configuración de vimrc a /etc:

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

Prepare Vim para la compilación:

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

Compilar el paquete:

```
make
```

Para preparar las pruebas, asegúrese de que el usuario tester pueda escribir en el árbol de código fuente y excluya un archivo que contenga pruebas que requieran **curl** o **wget**:

```
chown -R tester .
sed '/test_plugin_glvs/d' -i src/testdir/Make_all.mak
```

Ahora ejecute las pruebas como usuario tester:

```
su tester -c "TERM=xterm-256color LANG=en_US.UTF-8 make -j1 test" \
   &> vim-test.log
```

El conjunto de pruebas muestra una gran cantidad de datos binarios en la pantalla. Esto puede causar problemas con la configuración de la terminal actual (especialmente al sobrescribir la variable TERM para cumplir con algunas suposiciones del conjunto de pruebas). El problema se puede evitar redirigiendo la salida a un archivo de registro, como se muestra arriba. Una prueba exitosa mostrará el mensaje "ALL DONE" en el archivo de registro al finalizar.

Instalar el paquete:

```
make install
```

Muchos usuarios escriben "**vi**" en lugar de "**vim**" de forma automática. Para permitir la ejecución de **vim** cuando los usuarios escriben **vi** habitualmente, cree un enlace simbólico tanto para el binario como para la página del manual en los idiomas proporcionados:

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

Por defecto, la documentación de Vim se instala en /usr/share/vim. El siguiente enlace simbólico permite acceder a la documentación a través de /usr/share/doc/vim-9.1.1166, lo que la hace coherente con la ubicación de la documentación de otros paquetes:

```
ln -sv ../vim/vim91/doc /usr/share/doc/vim-9.1.1166
```

Si se va a instalar un sistema X Window en el sistema LFS, puede que sea necesario recompilar Vim después de instalar X. Vim incluye una versión GUI del editor que requiere la instalación de X y algunas bibliotecas adicionales. Para obtener más información sobre este proceso, consulte la documentación de Vim y la página de instalación de Vim en el libro de BLFS en https://www.linuxfromscratch.org/blfs/view/12.3/postlfs/vim.html.

8.73.2. Configuración de Vim

De forma predeterminada, **Vim** se ejecuta en modo incompatible con Vi. Esto puede resultar novedoso para los usuarios que ya han utilizado otros editores. La opción "nocompatible" se incluye a continuación para destacar que se está utilizando un nuevo comportamiento. También recuerda a quienes deseen cambiar al modo "compatible" que debe ser la primera opción en el archivo de configuración. Esto es necesario porque modifica otras opciones, y las modificaciones deben realizarse después de esta. Cree un archivo de configuración de **vim** predeterminado ejecutando lo siguiente:

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

" Asegúrese de que los valores predeterminados se configuren antes de personalizar
la configuración, no después.
source $VIMRUNTIME/defaults.vim
let skip_defaults_vim=1

set nocompatible
set backspace=2
set mouse=
syntax on
if (&term == "xterm") || (&term == "putty")
  set background=dark
endif

" Fin de /etc/vimrc
EOF
```

La opción "`set nocompatible`" hace que **vim** se comporte de forma más útil (por defecto) que con la compatibilidad con Vi.

Elimine el "no" para mantener el comportamiento anterior de **vi**. La opción "`set backspace=2`" permite retroceder en saltos de línea, sangrías automáticas y el inicio de una inserción. El parámetro "`syntax on`" habilita el resaltado de sintaxis de vim. La opción "`set mouse=`" permite pegar texto correctamente con el ratón al trabajar en chroot o mediante una conexión remota. Finalmente, la sentencia "if" con la opción "`set background=dark`" corrige la suposición de vim sobre el color de fondo de algunos emuladores de terminal. Esto proporciona al resaltado una mejor combinación de colores para su uso sobre el fondo negro de estos programas.

Puede obtener documentación sobre otras opciones disponibles ejecutando el siguiente comando:

```
vim -c ':options'
```

Nota

Por defecto, vim solo instala archivos de corrección ortográfica para El idioma inglés. Para instalar los archivos de corrección ortográfica de su idioma preferido, copie el archivo .spl y, opcionalmente, el archivo .sug correspondiente a su idioma y codificación de caracteres desde *runtime/spell* a */usr/share/vim/vim91/spell/*.

Para usar estos archivos de corrección ortográfica, se requiere cierta configuración en /etc/vimrc, por ejemplo:

```
set spelllang=en,ru
set spell
```

Para más información, consulte *runtime/spell/README.txt*.

8.73.3. Contenido de Vim

Programas instalados: ex (enlace a vim), rview (enlace a vim), rvim (enlace a vim), vi (enlace a vim), view (enlace a vim), vim, vimdiff (enlace a vim), vimtutor y xxd

Directorio de instalación: /usr/share/vim

Descripciones breves

ex Inicia **vim** en modo ex

rview Es una versión restringida de view; no se pueden iniciar comandos de shell ni suspender view.

rvim Es una versión restringida de vim; No se pueden iniciar comandos de shell y **vim** no se puede suspender.

vi Enlace a **vim**

view Inicia **vim** en modo de solo lectura.

vim Es el editor.

vimdiff Edita dos o tres versiones de un archivo con **vim** y muestra las diferencias.

vimtutor	Enseña las teclas y comandos básicos de **vim**.
xxd	Crea un volcado hexadecimal del archivo dado; también puede realizar la operación inversa, por lo que puede usarse para parches binarios.

8.74. MarkupSafe-3.0.2

MarkupSafe es un módulo de Python que implementa una cadena segura de marcado XML/HTML/XHTML.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 500 KB

8.74.1. Instalación de MarkupSafe

Compile MarkupSafe con el siguiente comando:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Este paquete no incluye un conjunto de pruebas.

Instale el paquete:

```
pip3 install --no-index --find-links dist Markupsafe
```

8.74.2. Contenido de MarkupSafe

Directorio de instalación: /usr/lib/python3.13/site-packages/MarkupSafe-3.0.2.dist-info

8.75. Jinja2-3.1.5

Jinja2 es un módulo de Python que implementa un lenguaje de plantillas Python simple.

Tiempo de compilación aproximado: Menos de 0.1 SBU
Espacio en disco requerido: 2.5 MB

8.75.1. Instalación de Jinja2

Compilación del paquete:

```
pip3 wheel -w dist --no-cache-dir --no-build-isolation --no-deps $PWD
```

Instalación del paquete:

```
pip3 install --no-index --find-links dist Jinja2
```

8.75.2. Contenido de Jinja2

Directorio de instalación: /usr/lib/python3.13/site-packages/Jinja2-3.1.5.dist-info

8.76. Udev desde Systemd-257.3

El paquete Udev contiene programas para la creación dinámica de nodos de dispositivos.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 161 MB

8.76.1. Instalación de Udev

Udev forma parte del paquete systemd-257.3. Utilice el archivo systemd-257.3.tar.xz como archivo tar de origen. Elimine dos grupos innecesarios, `render` y `sgx`, de las reglas predeterminadas de Udev:

```
sed -e 's/GROUP="render"/GROUP="video"/' \
    -e 's/GROUP="sgx", //'               \
    -i rules.d/50-udev-default.rules.in
```

Elimine una regla de Udev que requiere una instalación completa de Systemd:

```
sed -i '/systemd-sysctl/s/^/#/' rules.d/99-systemd.rules.in
```

Ajuste las rutas predefinidas de los archivos de configuración de red para la instalación independiente de Udev:

```
sed -e '/NETWORK_DIRS/s/systemd/udev/' \
    -i src/libsystemd/sd-network/network-util.h
```

Prepare Udev para la compilación:

```
mkdir -p build
cd       build
meson setup ..                  \
      --prefix=/usr             \
      --buildtype=release       \
      -D mode=release           \
      -D dev-kvm-mode=0660      \
      -D link-udev-shared=false \
      -D logind=false           \
      -D vconsole=false
```

Significado de las opciones de meson:

`--buildtype=release`

Esta opción anula el tipo de compilación predeterminado ("debug"), que produce binarios no optimizados.

`-D mode=release`

Desactiva algunas funciones consideradas experimentales por el desarrollador original.

-D dev-kvm-mode=0660

La regla predeterminada de udev permitiría a todos los usuarios acceder a `/dev/kvm`. Los editores la consideran peligrosa. Esta opción la anula.

-D link-udev-shared=false

Esta opción impide que udev se enlace a la biblioteca compartida interna de systemd, `libsystemd-shared`. Esta biblioteca está diseñada para ser compartida por varios componentes de Systemd y es demasiado compleja para una instalación exclusiva de udev.

-D logind=false -D vconsole=false

Estas opciones evitan la generación de varios archivos de reglas de udev pertenecientes a otros componentes de Systemd que no instalaremos.

Obtenga la lista de los ayudantes de udev incluidos y guárdela en una variable de entorno (exportarla no es estrictamente necesario, pero facilita la compilación como usuario normal o con un gestor de paquetes):

```
export udev_helpers=$(grep "'name' :" ../src/udev/meson.build | \
                      awk '{print $3}' | tr -d ",'" | grep -v 'udevadm')
```

Compile solo los componentes necesarios para udev:

```
ninja udevadm systemd-hwdb                                          \
      $(ninja -n | grep -Eo '(src/(lib)?udev|rules.d|hwdb.d)/[^ ]*') \
      $(realpath libudev.so --relative-to .)                        \
      $udev_helpers
```

Instalar el paquete:

```
install -vm755 -d {/usr/lib,/etc}/udev/{hwdb.d,rules.d,network}
install -vm755 -d /usr/{lib,share}/pkgconfig
install -vm755 udevadm                             /usr/bin/
install -vm755 systemd-hwdb                        /usr/bin/udev-hwdb
ln  -svfn ../bin/udevadm                           /usr/sbin/udevd
cp  -av  libudev.so{,*[0-9]}                       /usr/lib/
install -vm644 ../src/libudev/libudev.h            /usr/include/
install -vm644 src/libudev/*.pc                    /usr/lib/pkgconfig/
install -vm644 src/udev/*.pc                       /usr/share/pkgconfig/
install -vm644 ../src/udev/udev.conf               /etc/udev/
install -vm644 rules.d/* ../rules.d/README         /usr/lib/udev/rules.d/
install -vm644 $(find ../rules.d/*.rules \
                      -not -name '*power-switch*') /usr/lib/udev/rules.d/
install -vm644 hwdb.d/* ../hwdb.d/{*.hwdb,README}  /usr/lib/udev/hwdb.d/
install -vm755 $udev_helpers                       /usr/lib/udev
install -vm644 ../network/99-default.link          /usr/lib/udev/network
```

Instala algunas reglas personalizadas y archivos de soporte útiles en un entorno LFS:

```
tar -xvf ../../udev-lfs-20230818.tar.xz
make -f udev-lfs-20230818/Makefile.lfs install
```

Instala las páginas del manual:

```
tar -xf ../../systemd-man-pages-257.3.tar.xz                          \
    --no-same-owner --strip-components=1                              \
    -C /usr/share/man --wildcards '*/udev*' '*/libudev*'              \
                                  '*/systemd.link.5'                  \
                                  '*/systemd-'{hwdb,udevd.service}.8

sed 's|systemd/network|udev/network|'                                 \
    /usr/share/man/man5/systemd.link.5                                \
  > /usr/share/man/man5/udev.link.5

sed 's/systemd\(\\?-\)/udev\1/'  /usr/share/man/man8/systemd-hwdb.8 \
                               > /usr/share/man/man8/udev-hwdb.8

sed 's|lib.*udevd|sbin/udevd|'                                        \
    /usr/share/man/man8/systemd-udevd.service.8                       \
  > /usr/share/man/man8/udevd.8

rm /usr/share/man/man*/systemd*
```

Finalmente, desactive la variable udev_helpers:

```
unset udev_helpers
```

8.76.2. Configuración de Udev

La información sobre los dispositivos de hardware se guarda en los directorios /etc/udev/hwdb.d y /usr/lib/udev/hwdb.d. Udev necesita que dicha información se compile en una base de datos binaria llamada /etc/udev/hwdb.bin. Cree la base de datos inicial:

```
udev-hwdb update
```

Este comando debe ejecutarse cada vez que se actualice la información del hardware.

8.76.3. Contenido de Udev

Programas instalados: udevadm, udevd (enlace simbólico a udevadm) y udev-hwdb
Bibliotecas instaladas: libudev.so
Directorios instalados: /etc/udev y /usr/lib/udev

Descripciones breves

udevadm Herramienta genérica de administración de udev: controla el demonio udevd, proporciona información de la base de datos de Udev, supervisa los eventos UE, espera a que finalicen, prueba la configuración de Udev y los activa para un dispositivo determinado.

udevd Un demonio que detecta los eventos UE en el socket Netlink, crea dispositivos y ejecuta los programas externos configurados en respuesta a estos eventos UE.

udev-hwdb Actualiza o consulta la base de datos de hardware.

libudev Una interfaz de biblioteca para la información de dispositivos udev

/etc/udev Contiene archivos de configuración de Udev, permisos de dispositivos y reglas para la nomenclatura de dispositivos.

8.77. Man-DB-2.13.0

El paquete Man-DB contiene programas para buscar y visualizar páginas de manual.

Tiempo de compilación aproximado: 0.3 SBU
Espacio en disco requerido: 44 MB

8.77.1. Instalación de Man-DB

Preparar Man-DB para la compilación:

```
./configure --prefix=/usr                         \
            --docdir=/usr/share/doc/man-db-2.13.0 \
            --sysconfdir=/etc                     \
            --disable-setuid                      \
            --enable-cache-owner=bin              \
            --with-browser=/usr/bin/lynx          \
            --with-vgrind=/usr/bin/vgrind         \
            --with-grap=/usr/bin/grap             \
            --with-systemdtmpfilesdir=            \
            --with-systemdsystemunitdir=
```

Significado de las opciones de configuración:

`--disable-setuid`

Esto desactiva la asignación de setuid al programa man al usuario man.

`--enable-cache-owner=bin`

Esto cambia la propiedad de los archivos de caché del sistema al usuario bin.

`--with-...`

Estos tres parámetros se utilizan para configurar algunos programas predeterminados. Lynx es un navegador web basado en texto (consulte BLFS para obtener instrucciones de instalación), Vgrind convierte el código fuente del programa a la entrada de Groff y Grap es útil para la composición tipográfica de gráficos en documentos de Groff. Los programas Vgrind y Grap normalmente no son necesarios para ver las páginas del manual. No forman parte de LFS ni de BLFS, pero debería poder instalarlos usted mismo después de finalizar LFS si lo desea.

`--with-systemd...`

Estos parámetros evitan la instalación de directorios y archivos systemd innecesarios.

Compilar el paquete:

```
make
```

Para comprobar los resultados, ejecute:

```
make check
```

Instalar el paquete:

```
make install
```

8.77.2. Páginas del manual en otros idiomas en LFS

La siguiente tabla muestra el conjunto de caracteres con el que Man-DB asume que se codificarán las páginas del manual instaladas en /usr/share/man/<ll>. Además de esto, Man-DB determina correctamente si las páginas manuales instaladas en ese directorio están codificadas en UTF-8.

Tabla 8.1. Codificación de caracteres esperada de las páginas del manual de 8 bits heredadas

Idioma (código)	Codificación	Idioma (código)	Codificación
Danés (da)	ISO-8859-1	Croata (hr)	ISO-8859-2
Alemán (de)	ISO-8859-1	Húngaro (hu)	ISO-8859-2
Inglés (en)	ISO-8859-1	Japonés (ja)	EUC-JP
Español (es)	ISO-8859-1	Coreano (ko)	EUC-KR
Estonio (et)	ISO-8859-1	Lituano (lt)	ISO-8859-13
Finlandés (fi)	ISO-8859-1	Letón (lv)	ISO-8859-13
Francés (fr)	ISO-8859-1	Macedonio (mk)	ISO-8859-5
Irlandés (ga)	ISO-8859-1	Polaco (pl)	ISO-8859-2
Gallego (gl)	ISO-8859-1	Rumano (ro)	ISO-8859-2
Indonesio (id)	ISO-8859-1	Griego (el)	ISO-8859-7
Islandés (is)	ISO-8859-1	Eslovaco (sk)	ISO-8859-2
Italiano (it)	ISO-8859-1	Esloveno (sl)	ISO-8859-2
Bokmal noruego (nb)	ISO-8859-1	Serbio latino (sr@latin)	ISO-8859-2
Neerlandés (nl)	ISO-8859-1	Serbio (sr)	ISO-8859-5
Nynorsk noruego (nn)	ISO-8859-1	Turco (tr)	ISO-8859-9
Noruego (no)	ISO-8859-1	Ucraniano (uk)	KOI8-U
Portugués (pt)	ISO-8859-1	Vietnamita (vi)	TCVN5712-1
Sueco (sv)	ISO-8859-1	Chino simplificado (zh_CN)	GBK
Bielorruso (be)	CP1251	Chino simplificado de Singapur (zh_SG)	GBK
Búlgaro (bg)	CP1251	Chino tradicional de Hong Kong (zh_HK)	BIG5HKSCS
Checo (cs)	ISO-8859-2	Chino tradicional (zh_TW)	BIG5}

Nota

Las páginas del manual en idiomas no incluidos en la lista no son compatibles.

8.77.3. Contenido de Man-DB

Programas instalados: accessdb, apropos (enlace a whatis), catman, lexgrog, man, man-recode, mandb, manpath y whatis

Bibliotecas instaladas: libman.so y libmandb.so (ambas en /usr/lib/man-db)

Directorios instalados: /usr/lib/man-db, /usr/libexec/man-db y /usr/share/doc/man-db-2.13.0

Descripciones breves

accessdb Vuelca el contenido de la base de datos **whatis** en formato legible.

apropos Busca en la base de datos **whatis** y muestra descripciones breves de los comandos del sistema que contienen una cadena dada.

catman Crea o actualiza las páginas de manual preformateadas.

lexgrog Muestra información resumida de una línea sobre una página de manual dada.

man Formatea y muestra la página de manual solicitada.

man-recode Convierte las páginas de manual a otra codificación.

mandb Crea o actualiza la base de datos **whatis**.

manpath Muestra el contenido de $MANPATH o (si $MANPATH no está configurado) una ruta de búsqueda adecuada según la configuración de man.conf y el entorno del usuario.

whatis Busca en la base de datos **whatis** y muestra descripciones breves de los comandos del sistema que contienen la palabra clave dada como una palabra separada.

`libman` Contiene soporte en tiempo de ejecución para **man**.

`libmandb` Contiene soporte en tiempo de ejecución para **man**.

8.78. Procps-ng-4.0.5

El paquete Procps-ng contiene programas para monitorizar procesos.

Tiempo de compilación aproximado: 0.1 SBU
Espacio en disco requerido: 28 MB

8.78.1. Instalación de Procps-ng

Preparar Procps-ng para la compilación:

```
./configure --prefix=/usr                            \
            --docdir=/usr/share/doc/procps-ng-4.0.5 \
            --disable-static                         \
            --disable-kill                           \
            --enable-watch8bit
```

Significado de la opción de configuración:

`--disable-kill`

Esta opción deshabilita la compilación del comando **kill**; se instalará desde el paquete Util-linux.

`--enable-watch8bit`

Esta opción habilita la compatibilidad de ncursesw con el comando **watch**, de modo que pueda manejar caracteres de 8 bits.

Compilar el paquete:

```
make
```

Para ejecutar el conjunto de pruebas, ejecute:

```
chown -R tester .
su tester -c "PATH=$PATH make check"
```

Se sabe que una prueba llamada `ps con el indicador de salida bsdtime, cputime, etime, etimes` falla si el kernel del host no está compilado con `CONFIG_BSD_PROCESS_ACCT` habilitado. Además, una prueba pgrep puede fallar en el entorno chroot.

Instalar el paquete:

```
make install
```

8.78.2. Contenido de Procps-ng

Programas instalados: free, pgrep, pidof, pkill, pmap, ps, pwdx, slabtop, sysctl, tload, top, uptime, vmstat, w y watch

Biblioteca instalada: libproc-2.so

Directorios instalados: /usr/include/procps y /usr/share/doc/procps-ng-4.0.5

Descripciones breves

free	Informa sobre la cantidad de memoria libre y usada (física y de intercambio) en el sistema.
pgrep	Busca procesos según su nombre y otros atributos.
pidof	Informa sobre los PID de los programas.
pkill	Señaliza los procesos según su nombre y otros atributos.
pmap	Informa sobre el mapa de memoria del proceso.
ps	Enumera los procesos en ejecución
pwdx	Informa del directorio de trabajo actual de un proceso
slabtop	Muestra información detallada de la caché slab del núcleo en tiempo real
sysctl	Modifica los parámetros del núcleo en tiempo de ejecución
tload	Imprime un gráfico del promedio de carga actual del sistema
top	Muestra una lista de los procesos con mayor uso de la CPU; proporciona una visión continua de la actividad del procesador en tiempo real
uptime	Informa del tiempo de ejecución del sistema, la cantidad de usuarios conectados y los promedios de carga del sistema
vmstat	Informa de las estadísticas de memoria virtual, con información sobre procesos, memoria, paginación, entrada/salida (E/S) de bloques, traps y actividad de la CPU
w	Muestra qué usuarios han iniciado sesión, dónde y desde cuándo
watch	Ejecuta un comando repetidamente, mostrando la primera pantalla completa de su salida; esto permite al usuario observar cómo cambia la salida a lo largo del tiempo
`libproc-2`	Contiene las funciones utilizadas por la mayoría de los programas de este paquete

8.79. Util-linux-2.40.4

El paquete Util-linux contiene diversas utilidades. Entre ellas, se encuentran utilidades para gestionar sistemas de archivos, consolas, particiones y mensajes.

Tiempo de compilación aproximado: 0.5 SBU
Espacio en disco requerido: 316 MB

8.79.1. Instalación de Util-linux

Preparar Util-linux para la compilación:

```
./configure --bindir=/usr/bin     \
            --libdir=/usr/lib     \
            --runstatedir=/run    \
            --sbindir=/usr/sbin   \
            --disable-chfn-chsh   \
            --disable-login       \
            --disable-nologin     \
            --disable-su          \
            --disable-setpriv     \
            --disable-runuser     \
            --disable-pylibmount  \
            --disable-liblastlog2 \
            --disable-static      \
            --without-python      \
            --without-systemd     \
            --without-systemdsystemunitdir        \
            ADJTIME_PATH=/var/lib/hwclock/adjtime \
            --docdir=/usr/share/doc/util-linux-2.40.4
```

Las opciones --disable y --without evitan advertencias sobre la compilación de componentes que requieren paquetes que no están en LFS o que son incompatibles con los programas instalados por otros paquetes.

Compilar el paquete:

```
make
```

Si lo desea, cree un archivo *`/etc/fstab`* ficticio para realizar dos pruebas y ejecute el conjunto de pruebas como usuario no-*`root`*:

Advertencia

Ejecutar el conjunto de pruebas como usuario root puede ser perjudicial para el sistema. Para ejecutarlo, la opción CONFIG_SCSI_DEBUG del kernel debe estar disponible en el sistema en ejecución y compilarse como un módulo. Integrarla en el kernel impedirá el arranque. Para una cobertura completa, se deben instalar otros paquetes BLFS. Si se desea, esta prueba se puede ejecutar iniciando el sistema LFS completo y ejecutando:

```
bash tests/run.sh --srcdir=$PWD --builddir=$PWD
```

```
touch /etc/fstab
chown -R tester .
su tester -c "make -k check"
```

Las pruebas de enlace físico fallarán si el kernel del host no tiene habilitada la opción `CONFIG_CRYPTO_USER_API_HASH` o no tiene habilitada ninguna opción que proporcione una implementación `SHA256` (por ejemplo, `CONFIG_CRYPTO_SHA256` o `CONFIG_CRYPTO_SHA256_SSSE3` si la CPU admite SSE3 suplementario). Además, la prueba lsfd: inotify fallará si la opción del kernel `CONFIG_NETLINK_DIAG` no está habilitada.

Se sabe que otras dos pruebas, lsfd: columna SOURCE y utmp: last, fallan en el entorno chroot.

Instale el paquete:

```
make install
```

8.79.2. Contenido de Util-linux

Programas instalados: addpart, agetty, blkdiscard, blkid, blkzone, blockdev, cal, cfdisk, chcpu, chmem, choom, chrt, col, colcrt, colrm, column, ctrlaltdel, delpart, dmesg, eject, fallocate, fdisk, fincore, findfs, findmnt, flock, fsck, fsck.cramfs, fsck.minix, fsfreeze, fstrim, getopt, hardlink, hexdump, hwclock, i386 (enlace a setarch), ionice, ipcmk, ipcrm, ipcs, irqtop, isosize, kill, last, lastb (enlace a last), ldattach, linux32 (enlace a setarch), Linux64 (enlace a Setarch), registrador, look, losetup, lsblk, lscpu, lsipc, lsirq, lsfd, lslocks, lslogins, lsmem, lsns, mcookie, mesg, mkfs, mkfs.bfs, mkfs.cramfs, mkfs.minix, mkswap, more, mount, mountpoint, namei, nsenter, partx, pivot_root, prlimit, readprofile, rename, renice, resizepart, rev, rfkill, rtcwake, script, scriptlive, scriptreplay, setarch, setsid, setterm, sfdisk, sulogin, swaplabel, swapoff, swapon, switch_root, tasket, uclampset, ul, umount, uname26 (enlace a Setarch), dejar de compartir, utmpdump uuidd, uuidgen, uuidparse, wall, wdctl, whereis, wipefs, x86_64 (enlace a setarch) y zramctl

Bibliotecas: libblkid.so, libfdisk.so, libmount.so, libsmartcols.so y libuuid.so

Directorios: /usr/include/blkid, /usr/include/libfdisk, /usr/include/libmount, /usr/include/libsmartcols, /usr/include/uuid, /usr/share/doc/util-linux-2.40.4 y /var/lib/hwclock

Descripciones breves

addpart Informa al kernel de Linux sobre nuevas particiones.

agetty Abre un puerto TTY, solicita un nombre de usuario y luego invoca el programa de **inicio de sesión**.

blkdiscard Descarta sectores en un dispositivo.

blkid	Una utilidad de línea de comandos para localizar e imprimir atributos de dispositivos de bloque.
blkzone	Se utiliza para administrar dispositivos de bloque de almacenamiento por zonas.
blockdev	Permite a los usuarios llamar a las IOCTL de dispositivos de bloque desde la línea de comandos.
cal	Muestra un calendario simple.
cfdisk	Manipula la tabla de particiones del dispositivo dado.
chcpu	Modifica el estado de las CPU.
chmem	Configura la memoria.
choom	Muestra y ajusta las puntuaciones de OOM-killer, que se utilizan para determinar qué proceso eliminar primero cuando Linux se queda sin memoria.
chrt	Manipula los atributos en tiempo real de un proceso.
col	Filtra los avances de línea inversos.
colcrt	Filtra la salida **nroff** para terminales que carecen de algunas capacidades, como la sobrescritura y las medias líneas.
colrm	Filtra los datos dados. columnas
column	Formatea un archivo dado en varias columnas
ctrlaltsupr	Establece la función de la combinación de teclas Ctrl+Alt+Supr como restablecimiento completo o parcial.
delpart	Solicita al kernel de Linux que elimine una partición.
dmesg	Vuelca los mensajes de arranque del kernel.
eject	Expulsa los medios extraíbles.
fallocate	Preasigna espacio a un archivo.
fdisk	Manipula la tabla de particiones del dispositivo dado.
fincore	Cuenta las páginas de contenido de archivo en el núcleo.
findfs	Encuentra un sistema de archivos, ya sea por etiqueta o por Identificador Único Universal (UUID).
findmnt	Es una interfaz de línea de comandos para la biblioteca libmount para trabajar con archivos mountinfo, fstab y mtab.
flock	Adquiere un bloqueo de archivo y luego ejecuta un comando con el bloqueo activado.
fsck	Se utiliza para verificar y, opcionalmente, reparar sistemas de archivos.

fsck.cramfs	Realiza una comprobación de consistencia en el sistema de archivos Cramfs del dispositivo dado.
fsck.minix	Realiza una comprobación de consistencia en el sistema de archivos Minix del dispositivo dado.
fsfreeze	Es una envoltura muy simple para las operaciones del controlador de kernel ioctl FIFREEZE/FITHAW.
fstrim	Descarta los bloques no utilizados en un sistema de archivos montado.
getopt	Analiza las opciones en la línea de comandos dada.
hardlink	Consolida archivos duplicados mediante la creación de enlaces físicos.
hexdump	Vuelca el archivo dado en hexadecimal, decimal, octal o ascii
hwclock	Lee o configura el reloj de hardware del sistema, también llamado reloj de tiempo real (RTC) o reloj del sistema básico de entrada y salida (BIOS).
I386	Un enlace simbólico a setarch
ionice	Obtiene o establece la clase de programación de E/S y la prioridad de un programa.
ipcmk	Crea varios recursos IPC.
ipcrm	Elimina el recurso de Comunicación entre Procesos (IPC) especificado.
ipcs	Proporciona información del estado de IPC.
irqtop	Muestra la información del contador de interrupciones del núcleo en la vista superior (1).
isosize	Informa del tamaño de un sistema de archivos ISO9660.
kill	Envía señales a los procesos.
last	Muestra qué usuarios iniciaron (y cerraron) la última vez, buscando en el archivo /var/log/wtmp. También muestra los arranques, apagados y cambios en el nivel de ejecución del sistema.
lastb	Muestra los intentos fallidos de inicio de sesión, tal como se iniciaron en /var/log/btmp.
ldattach	Añade una disciplina de línea a una línea serie.
linux32	Un enlace simbólico a setarch.
linux64	Un enlace simbólico a setarch.
logger	Introduce el mensaje especificado en el registro del sistema.
look	Muestra las líneas que comienzan con la cadena especificada.
losetup	Configura y controla dispositivos de bucle.
lsblk	Muestra información sobre todos los dispositivos de bloque o algunos seleccionados en un formato de árbol.

lscpu	Imprime información de la arquitectura de la CPU.
lsfd	Muestra información sobre los archivos abiertos; reemplaza a **lsof**.
lsipc	Imprime información sobre las funciones IPC actualmente empleadas en el sistema.
lsirq	Muestra información del contador de interrupciones del núcleo.
lslocks	Enumera los bloqueos del sistema local.
lslogins	Enumera información sobre usuarios, grupos y cuentas del sistema.
lsmem	Enumera los rangos de memoria disponible con su estado en línea.
lsns	Enumera los espacios de nombres.
mcookie	Genera cookies mágicas (números hexadecimales aleatorios de 128 bits) para **xauth**.
mesg	Controla si otros usuarios pueden enviar mensajes al terminal del usuario actual.
mkfs	Construye un sistema de archivos en un dispositivo (normalmente una partición del disco duro).
mkfs.bfs	Crea un sistema de archivos bfs de Santa Cruz Operations (SCO).
mkfs.cramfs	Crea un sistema de archivos cramfs.
mkfs.minix	Crea un sistema de archivos Minix.
mkswap	Inicializa el dispositivo o archivo dado para usarlo como área de intercambio.
more	Un filtro para navegar por el texto pantalla por pantalla.
mount	Conecta el sistema de archivos del dispositivo dado a un directorio específico en el árbol del sistema de archivos.
mountpoint	Comprueba si el directorio es un punto de montaje.
namei	Muestra los enlaces simbólicos en las rutas dadas.
nsenter	Ejecuta un programa con los espacios de nombres de otros procesos.
partx	Informa al núcleo sobre la presencia y numeración de las particiones en disco.
pivot_root	Convierte el sistema de archivos dado en el nuevo sistema de archivos raíz del proceso actual.
prlimit	Obtiene y establece los límites de recursos de un proceso.
readprofile	Lee la información de perfil del núcleo.
rename	Renombra los archivos dados, reemplazando una cadena dada por otra.
renice	Modifica la prioridad de los procesos en ejecución.
resizepart	Solicita al núcleo de Linux que redimensione una partición.
rev	Invierte las líneas de un archivo dado.

rfkill	Herramienta para habilitar y deshabilitar dispositivos inalámbricos.
rtcwake	Se utiliza para entrar en un estado de suspensión del sistema hasta la hora de activación especificada.
script	Crea un script de una sesión de terminal.
scriptlive	Reejecuta scripts de sesión usando información de tiempo.
scriptreplay	Reproduce scripts de sesión usando información de tiempo.
setarch	Cambia la arquitectura reportada en un nuevo entorno de programa y establece indicadores de personalidad.
setsid	Ejecuta el programa dado en una nueva sesión.
setterm	Establece los atributos de la terminal
sfdisk	Un manipulador de la tabla de particiones de disco
sulogin	Permite al usuario root iniciar sesión; normalmente lo invoca init cuando el sistema entra en modo monousuario
swaplabel	Realiza cambios en el UUID y la etiqueta del área de intercambio
swapoff	Desactiva la paginación y el intercambio de dispositivos y archivos
swapon	Habilita la paginación y el intercambio de dispositivos y archivos, y lista los dispositivos y archivos actualmente en uso
switch_root	Cambia a otro sistema de archivos como raíz del árbol de montaje
taskset	Recupera o establece la afinidad de CPU de un proceso
uclampset	Manipula los atributos de restricción de utilización del sistema o de un proceso
ul	Un filtro para traducir guiones bajos en secuencias de escape que indican subrayado para la terminal en uso
umount	Desconecta un sistema de archivos del árbol de archivos del sistema
uname26	Un enlace simbólico a setarch
unshare	Ejecuta un programa con algunos espacios de nombres no compartidos con el padre
utmpdump	Muestra el contenido del archivo de inicio de sesión dado en un formato intuitivo
uuidd	Un demonio utilizado por la biblioteca UUID para generar UUID basados en tiempo de forma segura y con garantía de unicidad
uuidgen	Crea nuevos UUID. Cada nuevo UUID es un número aleatorio que probablemente sea único entre todos los UUID creados, tanto en el sistema local como en otros sistemas, en el pasado y en el futuro, con una probabilidad extremadamente alta (2 UUID son posibles).

uuidparse	Una utilidad para analizar identificadores únicos.
wall	Muestra el contenido de un archivo o, por defecto, su entrada estándar, en las terminales de todos los usuarios conectados.
wdctl	Muestra el estado del sistema de vigilancia del hardware.
whereis	Informa de la ubicación de los archivos binarios, fuente y de la página de manual del comando dado.
wipefs	Borra la firma del sistema de archivos de un dispositivo.
x86_64	Un enlace simbólico a setarch.
zramctl	Un programa para configurar y controlar dispositivos zram (disco RAM comprimido).
`libblkid`	Contiene rutinas para la identificación de dispositivos y la extracción de tokens.
`libfdisk`	Contiene rutinas para manipular tablas de particiones.
`libmount`	Contiene rutinas para el montaje y desmontaje de dispositivos de bloque.
`libsmartcols`	Contiene rutinas para facilitar la salida de pantalla en formato tabular.
`libuuid`	Contiene rutinas para generar identificadores únicos para objetos accesibles más allá del sistema local.

8.80. E2fsprogs-1.47.2

El paquete E2fsprogs contiene las utilidades para gestionar el sistema de archivos ext2. También es compatible con los sistemas de archivos con registro en diario ext3 y ext4.

Tiempo de compilación aproximado: 2,4 SBU en un disco duro, 0,5 SBU en un SSD
Espacio en disco requerido: 99 MB

8.80.1. Instalación de E2fsprogs

La documentación de E2fsprogs recomienda compilar el paquete en un subdirectorio del árbol de código fuente:

```
mkdir -v build
cd       build
```

Preparar E2fsprogs para la compilación:

```
../configure --prefix=/usr        \
             --sysconfdir=/etc    \
             --enable-elf-shlibs  \
             --disable-libblkid   \
             --disable-libuuid    \
             --disable-uuidd      \
             --disable-fsck
```

Significado de las opciones de configuración:

`--enable-elf-shlibs`

Esto crea las bibliotecas compartidas que utilizan algunos programas de este paquete.

`--disable-*`

Esto impide compilar e instalar las bibliotecas `libuuid` y `libblkid`, el demonio `uuidd` y el contenedor **fsck**; util-linux instala versiones más recientes. Compilar el paquete:

```
make
```

Para ejecutar las pruebas, ejecute:

```
make check
```

Se sabe que una prueba llamada `m_assume_storage_prezeroed` falla. Otra prueba llamada `m_rootdir_acl` falla si el sistema de archivos utilizado para el sistema LFS no es `ext4`.

Instalar el paquete:

```
make install
```

Eliminar las bibliotecas estáticas innecesarias:

```
rm -fv /usr/lib/{libcom_err,libe2p,libext2fs,libss}.a
```

Este paquete instala un archivo `.info` comprimido con gzip, pero no actualiza el archivo `dir` del sistema. Descomprima este archivo y actualice el archivo `dir` del sistema con los siguientes comandos:

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

Si lo desea, cree e instale documentación adicional con los siguientes comandos:

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

8.80.2. Configuración de E2fsprogs

`/etc/mke2fs.conf` contiene los valores predeterminados de varias opciones de línea de comandos de **mke2fs**. Puede editar el archivo para que los valores predeterminados se ajusten a sus necesidades. Por ejemplo, algunas utilidades (no en LFS ni BLFS) no reconocen un sistema de archivos `ext4` con la función `metadata_csum_seed` habilitada. **Si** necesita dicha utilidad, puede eliminar la función de la lista de funciones predeterminadas de `ext4` con el comando:

```
sed 's/metadata_csum_seed,//' -i /etc/mke2fs.conf
```

Consulte la página del manual *mke2fs.conf(5)* para obtener más información.

8.80.3. Contenido de E2fsprogs

Programas instalados: badblocks, chattr, compile_et, debugfs, dumpe2fs, e2freefrag, e2fsck, e2image, e2label, e2mmpstatus, e2scrub, e2scrub_all, e2undo, e4crypt, e4defrag, filefrag, fsck.ext2, fsck.ext3, fsck.ext4, logsave, lsattr, mk_cmds, mke2fs, mkfs.ext2, mkfs.ext3, mkfs.ext4, mklost+found, resize2fs y tune2fs

Bibliotecas instaladas: libcom_err.so, libe2p.so, libext2fs.so y libss.so

Directorios instalados: /usr/include/e2p, /usr/include/et, /usr/include/ext2fs, /usr/include/ss, /usr/lib/e2fsprogs, /usr/share/et y /usr/share/ss

Descripciones breves

badblocks Busca bloques defectuosos en un dispositivo (normalmente una partición de disco).

chattr Cambia los atributos de los archivos en sistemas de archivos *`ext{234}`*.

compile_et Un compilador de tablas de errores; convierte una tabla de nombres y mensajes de códigos de error en un archivo fuente en C. Ideal para usar con la biblioteca *`com_err`*.

debugfs Un depurador de sistemas de archivos. Se puede usar para examinar y cambiar el estado de los sistemas de archivos *`ext{234}`*.

dumpe2fs Imprime la información del superbloque y del grupo de bloques del sistema de archivos presente en un dispositivo.

e2freefrag Informa sobre la fragmentación del espacio libre.

e2fsck Se usa para comprobar y, opcionalmente, reparar sistemas de archivos *ext{234}*.

e2image Se usa para guardar datos críticos del sistema de archivos *ext{234}* en un archivo.

e2label Muestra o cambia la etiqueta del sistema de archivos *ext{234}* en un dispositivo.

e2mmpstatus Comprueba el estado de MMP (Protección de Montaje Múltiple) de un sistema de archivos ext4.

e2scrub Comprueba el contenido de un sistema de archivos *ext{234}* montado.

e2scrub_all Comprueba si hay errores en todos los sistemas de archivos *ext{234}* montados.

e2undo Reproduce el registro de deshacer de un sistema de archivos *ext{234}* encontrado en un dispositivo. [Esto puede usarse para deshacer una operación fallida de un programa E2fsprogs.]

e4crypt Utilidad de cifrado del sistema de archivos *Ext4*

e4defrag Desfragmentador en línea para sistemas de archivos e*xt4*

filefrag Informa sobre el grado de fragmentación de un archivo en particular.

fsck.ext2 Por defecto, verifica los sistemas de archivos ext2 y es un enlace duro a **e2fsck**.

fsck.ext3 Por defecto, verifica los sistemas de archivos ext3 y es un enlace duro a **e2fsck**.

fsck.ext4 Por defecto, verifica los sistemas de archivos ext4 y es un enlace duro a **e2fsck**.

logsave Guarda la salida de un comando en un archivo de registro.

lsattr Enumera los atributos de los archivos en un segundo sistema de archivos extendido.

mk_cmds Convierte una tabla de nombres de comandos y mensajes de ayuda en un archivo fuente en C compatible con la biblioteca del subsistema libss.

mke2fs Crea un sistema de archivos ext{234} en el dispositivo indicado.

mkfs.ext2 Por defecto, crea sistemas de archivos *ext2* y es un enlace duro a **mke2fs**.

mkfs.ext3 Por defecto, crea sistemas de archivos *ext3* y es un enlace duro a **mke2fs**.

mkfs.ext4 Por defecto, crea sistemas de archivos *ext4* y es un enlace duro a **mke2fs**.

mklost+found Crea un directorio *lost+found* en un sistema de archivos *ext{234}*. Preasigna bloques de disco a este directorio para simplificar la tarea de **e2fsck**.

resize2fs Puede usarse para ampliar o reducir el tamaño de los sistemas de archivos *ext{234}*.

tune2fs Ajusta los parámetros configurables del sistema de archivos en sistemas de archivos *ext{234}*.

`libcom_err` La rutina común de visualización de errores.

`libe2p` Usada por **dumpe2fs**, **chattr** y **lsattr**.

`libext2fs` Contiene rutinas que permiten a los programas de usuario manipular sistemas de archivos *ext{234}*.

`libss` Usada por **debugfs**.

8.81. Sysklogd-2.7.0

El paquete Sysklogd contiene programas para registrar mensajes del sistema, como los que emite el núcleo cuando ocurren eventos inusuales.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 4,1 MB

8.81.1. Instalación de Sysklogd

Preparar el paquete para la compilación:

```
./configure --prefix=/usr       \
            --sysconfdir=/etc   \
            --runstatedir=/run  \
            --without-logger    \
            --disable-static    \
            --docdir=/usr/share/doc/sysklogd-2.7.0
```

Compilar el paquete:

```
make
```

Este paquete no incluye un conjunto de pruebas.

Instalar el paquete:

```
make install
```

8.81.2. Configuración de Sysklogd

Cree un nuevo archivo /etc/syslog.conf ejecutando lo siguiente:

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

auth,authpriv.* -/var/log/auth.log
*.*;auth,authpriv.none -/var/log/sys.log
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* -/var/log/mail.log
user.* -/var/log/user.log
*.emerg *

# Do not open any internet ports.
secure_mode 2

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

8.81.3. Contenido de Sysklogd

Programa instalado: syslogd

Descripciones breves

syslogd Registra los mensajes que los programas del sistema ofrecen para su registro. [Cada mensaje registrado contiene al menos una marca de fecha y un nombre de host, y normalmente también el nombre del programa, pero esto depende de la confianza que se le indique al demonio de registro].

8.82. SysVinit-3.14

El paquete SysVinit contiene programas para controlar el inicio, la ejecución y el apagado del sistema.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 2,9 MB

8.82.1. Instalación de SysVinit

Primero, aplique un parche que elimina varios programas instalados por otros paquetes, aclara un mensaje y corrige una advertencia del compilador:

```
patch -Np1 -i ../sysvinit-3.14-consolidated-1.patch
```

Compilar el paquete:

```
make
```

Este paquete no incluye un conjunto de pruebas.

Instalar el paquete:

```
make install
```

8.82.2. Contenido de SysVinit

Programas instalados: bootlogd, fstab-decode, halt, init, killall5, poweroff (enlace a halt), reboot (enlace a halt), runlevel, shutoff y telinit (enlace a init)

Descripciones breves

bootlogd Registra los mensajes de arranque en un archivo de registro.

fstab-decode Ejecuta un comando con argumentos codificados en fstab.

halt Normalmente invoca el **apagado** con la opción `-h`, pero cuando ya está en el nivel de ejecución 0, le indica al núcleo que detenga el sistema; anota en el archivo *`/var/log/wtmp`* que el sistema se está apagando.

init El primer proceso que se inicia cuando el núcleo ha inicializado el hardware; asume el proceso de arranque e inicia todos los procesos especificados en su archivo de configuración.

killall5 Envía una señal a todos los procesos, excepto a los de su propia sesión. No eliminará su shell principal.

poweroff Indica al kernel que detenga el sistema y apague el ordenador (ver **halt**).

reboot Indica al kernel que reinicie el sistema (ver **halt**).

runlevel Informa del nivel de ejecución anterior y actual, como se indica en el último registro de nivel de ejecución en */run/utmp*.

shutdown Detiene el sistema de forma segura, enviando señales a todos los procesos y notificando a todos los usuarios conectados.

telinit Indica a **init** a qué nivel de ejecución cambiar.

8.83. Acerca de los símbolos de depuración

La mayoría de los programas y bibliotecas se compilan, por defecto, con símbolos de depuración incluidos (con la opción `-g` de gcc). Esto significa que, al depurar un programa o biblioteca compilado con información de depuración, el depurador puede proporcionar no solo las direcciones de memoria, sino también los nombres de las rutinas y las variables.

La inclusión de estos símbolos de depuración amplía significativamente el programa o la biblioteca. A continuación, se muestran dos ejemplos del espacio que ocupan estos símbolos:

• Un binario **bash** con símbolos de depuración: 1200 KB

• Un binario **bash** sin símbolos de depuración: 480 KB (60 % menor)

• Archivos Glibc y GCC (*/lib* y */usr/lib*) con símbolos de depuración: 87 MB

• Archivos Glibc y GCC sin símbolos de depuración: 16 MB (82 % menor)

El tamaño varía según el compilador y la biblioteca de C utilizados, pero un programa al que se le han eliminado los símbolos de depuración suele ser entre un 50 % y un 80 % menor que su equivalente sin ellos. Dado que la mayoría de los usuarios nunca utilizarán un depurador en el software de su sistema, se puede recuperar mucho espacio en disco eliminando estos símbolos. La siguiente sección muestra cómo eliminar todos los símbolos de depuración de los programas y bibliotecas.

8.84. Eliminación

Esta sección es opcional. Si el usuario previsto no es programador y no planea depurar el software del sistema, el tamaño del sistema se puede reducir en unos 2 GB eliminando los símbolos de depuración y algunas entradas innecesarias de la tabla de símbolos de los binarios y bibliotecas. Esto no supone ningún inconveniente para un usuario típico de Linux.

La mayoría de las personas que utilizan los comandos mencionados a continuación no experimentan ninguna dificultad. Sin embargo, es fácil cometer un error y dejar el nuevo sistema inutilizable. Por lo tanto, antes de ejecutar los comandos de **depuración** (**strip**), conviene realizar una copia de seguridad del sistema LFS en su estado actual.

Un comando de **depuración** con la opción *--strip-unneeded* elimina todos los símbolos de depuración de un binario o biblioteca. También elimina todas las entradas de la tabla de símbolos que no necesita el enlazador (para bibliotecas estáticas) o el enlazador dinámico (para binarios enlazados dinámicamente y bibliotecas compartidas).

Los símbolos de **depuración** de las bibliotecas seleccionadas se comprimen con Zlib y se conservan en archivos separados. Esa información de depuración es necesaria para ejecutar pruebas de regresión con `valgrind` o `gdb` posteriormente, en BLFS.

Tenga en cuenta que **depurar** (**strip**) sobrescribirá el archivo binario o de biblioteca que esté procesando. Esto puede bloquear los procesos que usan código o datos del archivo. Si el proceso que ejecuta strip se ve afectado, el binario o la biblioteca que se está desmontando puede destruirse; esto

puede dejar el sistema completamente inutilizable. Para evitar este problema, copiamos algunas bibliotecas y binarios en `/tmp`, los desmontamos allí y luego los reinstalamos con el comando **install**. (La entrada relacionada en la Sección 8.2.1, "Problemas de actualización", explica la justificación para usar el comando **install** aquí).

Nota

El nombre del cargador ELF es l*d-linux-x86-64.so.2* en sistemas de 64 bits y ld-linux.so.2 en sistemas de 32 bits. La **siguiente** construcción selecciona el nombre correcto para la arquitectura actual, excluyendo cualquier cosa que termine en g, en caso de que los siguientes comandos ya se hayan ejecutado.

Importante

Si hay algún paquete cuya versión es diferente a la especificada en el libro (ya sea por un aviso de seguridad o por gusto personal), podría ser necesario actualizar el nombre del archivo de la biblioteca en *save_usrlib* o *online_usrlib*. **De lo contrario, el sistema podría quedar completamente inutilizable**.

```
save_usrlib="$(cd /usr/lib; ls ld-linux*[^g])
             libc.so.6
             libthread_db.so.1
             libquadmath.so.0.0.0
             libstdc++.so.6.0.33
             libitm.so.1.0.0
             libatomic.so.1.2.0"
cd /usr/lib

for LIB in $save_usrlib; do
    objcopy --only-keep-debug --compress-debug-sections=zlib $LIB $LIB.dbg
    cp $LIB /tmp/$LIB
    strip --strip-unneeded /tmp/$LIB
    objcopy --add-gnu-debuglink=$LIB.dbg /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

online_usrbin="bash find strip"
online_usrlib="libbfd-2.44.so
               libsframe.so.1.0.0
               libhistory.so.8.2
               libncursesw.so.6.5
               libm.so.6
               libreadline.so.8.2
               libz.so.1.3.1
               libzstd.so.1.5.7
               $(cd /usr/lib; find libnss*.so* -type f)"

for BIN in $online_usrbin; do
    cp /usr/bin/$BIN /tmp/$BIN
    strip --strip-unneeded /tmp/$BIN
    install -vm755 /tmp/$BIN /usr/bin
    rm /tmp/$BIN
done

for LIB in $online_usrlib; do
    cp /usr/lib/$LIB /tmp/$LIB
```

```
    strip --strip-unneeded /tmp/$LIB
    install -vm755 /tmp/$LIB /usr/lib
    rm /tmp/$LIB
done

for i in $(find /usr/lib -type f -name \*.so* ! -name \*dbg)  \
         $(find /usr/lib -type f -name \*.a)                  \
         $(find /usr/{bin,sbin,libexec} -type f); do
    case "$online_usrbin $online_usrlib $save_usrlib" in
        *$(basename $i)* )
            ;;
        * ) strip --strip-unneeded $i
            ;;
    esac
done

unset BIN LIB save_usrlib online_usrbin online_usrlib
```

Una gran cantidad de archivos se marcarán como errores porque no se reconoce su formato. Estas advertencias se pueden ignorar sin problemas. Indican que esos archivos son scripts, no binarios.

8.85. Limpieza

Finalmente, limpie algunos archivos sobrantes de las pruebas:

```
rm -rf /tmp/{*,.*}
```

También hay varios archivos en los directorios /usr/lib y /usr/libexec con la extensión .la. Estos son archivos "libtool". En un sistema Linux moderno, los archivos .la de libtool solo son útiles para libltdl. No se espera que libltdl cargue ninguna biblioteca en LFS, y se sabe que algunos archivos .la pueden interrumpir la compilación de paquetes BLFS. Elimine esos archivos ahora:

```
find /usr/lib /usr/libexec -name \*.la -delete
```

Para obtener más información sobre los archivos comprimidos de libtool, consulte la sección de *BLFS "Acerca de los archivos comprimidos de libtool (.la)*".

El compilador creado en los capítulos 6 y 7 aún está parcialmente instalado y ya no es necesario. Elimínelo con:

```
find /usr -depth -name $(uname -m)-lfs-linux-gnu\* | xargs rm -rf
```

Finalmente, elimine la cuenta de usuario temporal "tester" creada al principio del capítulo anterior.

```
userdel -r tester
```

Capítulo 9. Configuración del sistema

9.1. Introducción

Arrancar un sistema Linux implica varias tareas. El proceso debe montar sistemas de archivos virtuales y reales, inicializar dispositivos, comprobar la integridad de los sistemas de archivos, montar y activar particiones o archivos de intercambio, configurar el reloj del sistema, activar la red, iniciar los daemons requeridos por el sistema y realizar cualquier otra tarea personalizada especificada por el usuario. Este proceso debe organizarse para garantizar que las tareas se realicen en el orden correcto y se ejecuten lo más rápido posible.

9.1.1. System V

System V es el proceso de arranque clásico que se ha utilizado en Unix y sistemas similares a Unix, como Linux, desde aproximadamente 1983. Consiste en un pequeño programa, init, que configura procesos básicos como el **inicio de sesión** (mediante getty) y ejecuta un script. Este script, generalmente llamado rc, controla la ejecución de un conjunto de scripts adicionales que realizan las tareas necesarias para inicializar el sistema.

El programa **init** se controla mediante el archivo */etc/inittab* y se organiza en niveles de ejecución que el usuario puede seleccionar.

En LFS, se utilizan de la siguiente manera:

```
0 – detener (halt)
1 – Modo monousuario (Single user mode)
2 – Definible por el usuario (User definable)
3 – Modo multiusuario completo (Full multiuser mode)
4 – Definible por el usuario (User definable)
5 – Modo multiusuario completo con administrador de pantalla (Full multiuser mode
    with display manager)
6 – reiniciar (reboot)
```

El nivel de ejecución predeterminado habitual es 3 o 5.

Ventajas

• Sistema establecido y bien comprendido.

• Fácil de personalizar.

Desventajas

• Puede ser más lento al arrancar. Un sistema LFS básico de velocidad media tarda entre 8 y 12 segundos, donde el tiempo de arranque se mide desde el primer mensaje del kernel hasta la solicitud de inicio de sesión. La conectividad de red suele establecerse unos 2 segundos después de la solicitud de inicio de sesión.

• Procesamiento en serie de las tareas de arranque. Esto está relacionado con el punto anterior. Un retraso en cualquier proceso, como una comprobación del sistema de archivos, retrasará todo el proceso de arranque.

• No admite directamente funciones avanzadas como grupos de control (cgroups) ni programación de reparto equitativo por usuario.

• Agregar scripts requiere decisiones de secuenciación estáticas y manuales.

9.2. LFS-Bootscripts-20240825

El paquete LFS-Bootscripts contiene un conjunto de scripts para iniciar/detener el sistema LFS al arrancar/apagar. Los archivos de configuración y los procedimientos necesarios para personalizar el proceso de arranque se describen en las siguientes secciones.

Tiempo de compilación aproximado: Menos de 0,1 SBU
Espacio en disco requerido: 244 KB

9.2.1. Instalación de LFS-Bootscripts

Instalar el paquete:

```
make install
```

9.2.2. Contenido de los scripts de arranque de LFS

Scripts instalados: checkfs, cleanfs, console, functions, halt, ifdown, ifup, localnet, modules, mountfs, mountvirtfs, network, rc, reboot, sendsignals, setclock, ipv4-static, swap, sysctl, sysklogd, template, udev y udev_retry

Directorios instalados: /etc/rc.d, /etc/init.d (enlace simbólico), /etc/sysconfig, /lib/services, /lib/lsb (enlace simbólico)

Descripciones breves

checkfs Comprueba la integridad de los sistemas de archivos antes de montarlos (con excepción de los sistemas de archivos de registro y basados en red).

cleanfs Elimina los archivos que no deben conservarse entre reinicios, como los de *`/run/`* y *`/var/lock/`*; recrea *`/run/utmp`* y elimina los archivos *`/etc/nologin`*, */fastboot* y */forcefsck* posiblemente presentes.

console Carga la tabla de mapas de teclas correcta para la distribución de teclado deseada. También establece la fuente de la pantalla.

functions Contiene funciones comunes, como la comprobación de errores y estado, que utilizan varios scripts de arranque.

halt Detiene el sistema.

ifdown Detiene un dispositivo de red.

ifup Inicializa un dispositivo de red.

localnet Configura el nombre de host del sistema y el dispositivo de bucle invertido local.

modules Carga los módulos del kernel listados en *`/etc/sysconfig/modules`*, utilizando los argumentos que también se proporcionan allí.

mountfs Monta todos los sistemas de archivos, excepto aquellos que están marcados como *noauto* o que están basados en red.

mountvirtfs Monta sistemas de archivos virtuales del kernel, como `proc`.

network Configura interfaces de red, como tarjetas de red, y configura la puerta de enlace predeterminada (cuando corresponda).

rc El script de control de nivel de ejecución maestro. Es responsable de ejecutar todos los demás scripts de arranque uno por uno, en una secuencia determinada por los nombres de los enlaces simbólicos a esos otros scripts de arranque.

reboot Reinicia el sistema.

sendsignals Se asegura de que todos los procesos finalicen antes de que el sistema se reinicie o se detenga.

setclock Reinicia el reloj del sistema a la hora local si el reloj del hardware no está configurado en UTC.

ipv4-static Proporciona la funcionalidad necesaria para asignar una dirección IP estática a una interfaz de red.

swap Habilita y deshabilita archivos de intercambio y particiones.

sysctl Carga los valores de configuración del sistema desde `/etc/sysctl.conf`, si existe dicho archivo, en el kernel en ejecución.

sysklogd Inicia y detiene los daemons de registro del sistema y del kernel.

template Una plantilla para crear scripts de arranque personalizados para otros daemons.

udev Prepara el directorio `/dev` e inicia el daemon udev.

udev_retry Reintenta los eventos udev fallidos y copia los archivos de reglas generados desde `/run/udev` a `/etc/udev/rules.d` si es necesario.

9.3. Resumen del manejo de dispositivos y módulos

En el capítulo 8, instalamos el demonio udev durante la compilación de udev. Antes de profundizar en el funcionamiento de udev, conviene repasar brevemente los métodos anteriores de manejo de dispositivos.

Los sistemas Linux, en general, utilizaban tradicionalmente un método estático de creación de dispositivos, mediante el cual se creaban numerosos nodos de dispositivo bajo `/dev` (a veces, literalmente miles de nodos), independientemente de si los dispositivos de hardware correspondientes existían. Esto se hacía normalmente mediante un script **MAKEDEV**, que contenía varias llamadas al programa **mknod** con los números de dispositivo mayor y menor relevantes para cada dispositivo posible existente.

Con el método udev, solo se crean nodos de dispositivo para aquellos dispositivos detectados por el kernel. Estos nodos de dispositivo se crean cada vez que se inicia el sistema y se almacenan en un sistema de archivos `devtmpfs` (un sistema de archivos virtual que reside completamente en la memoria del sistema). Los nodos de dispositivo no requieren mucho espacio, por lo que la memoria utilizada es insignificante.

9.3.1. Historia

En febrero de 2000, un nuevo sistema de archivos llamado *`devfs`* se integró en el kernel 2.3.46 y estuvo disponible durante la serie 2.4 de kernels estables. Aunque estaba presente en el código fuente del kernel, este método de creación dinámica de dispositivos nunca recibió un apoyo abrumador por parte de los desarrolladores principales del kernel.

El principal problema con el enfoque adoptado por *`devfs`* era la forma en que gestionaba la detección, creación y asignación de nombres de dispositivos.

Este último problema, el de la asignación de nombres a los nodos de dispositivo, fue quizás el más crítico. Se acepta generalmente que, si los nombres de dispositivo son configurables, la política de asignación de nombres de dispositivos debe ser elegida por los administradores del sistema, y no impuesta por los desarrolladores. El sistema de archivos *`devfs`* también presentaba condiciones de carrera inherentes a su diseño; estas no se podían solucionar sin una revisión a fondo del kernel. *`devfs`* estuvo marcado como obsoleto durante mucho tiempo y finalmente se eliminó del kernel en junio de 2006.

Con el desarrollo del árbol de kernel inestable 2.5, posteriormente lanzado como la serie 2.6 de kernels estables, surgió un nuevo sistema de archivos virtual llamado *`sysfs`*. La función de *`sysfs`* es proporcionar información sobre la configuración del hardware del sistema a los procesos del espacio de usuario. Con esta representación visible en el espacio de usuario, fue posible desarrollar un reemplazo para *`devfs`* en el espacio de usuario.

9.3.2. Implementación de Udev

9.3.2.1. Sysfs

El sistema de archivos *sysfs* se mencionó brevemente anteriormente. Cabe preguntarse cómo sysfs conoce los dispositivos presentes en un sistema y qué números de dispositivo deben usarse para ellos. Los controladores compilados en el kernel registran sus objetos en *sysfs* (*devtmpfs* internamente) a medida que el kernel los detecta. Para los controladores compilados como módulos, el registro se realiza al cargar el módulo. Una vez montado el sistema de archivos *sysfs* (en */sys*), los datos que los controladores han registrado con *sysfs* están disponibles para los procesos del espacio de usuario y para udevd para su procesamiento (incluidas las modificaciones en los nodos de dispositivo).

9.3.2.2. Creación de nodos de dispositivo

El núcleo crea los archivos de dispositivo en el sistema de archivos *devtmpfs*. Cualquier controlador que desee registrar un nodo de dispositivo utilizará *devtmpfs* (a través del núcleo del controlador) para hacerlo. Cuando una instancia de *devtmpfs* se monta en */dev*, el nodo de dispositivo se expone inicialmente al espacio de usuario con un nombre, permisos y propietario fijos.

Poco después, el núcleo envía un uevent a **udevd**. Según las reglas especificadas en los archivos dentro de los directorios */etc/udev/rules.d*, */usr/lib/udev/rules.d* y */run/udev/rules.d*, **udevd** creará enlaces simbólicos adicionales al nodo del dispositivo, o cambiará sus permisos, propietario o grupo, o modificará la entrada interna de la base de datos de **udevd** (nombre) para ese objeto.

Las reglas de estos tres directorios están numeradas y se fusionan. Si **udevd** no encuentra una regla para el dispositivo que está creando, conservará los permisos y la propiedad del archivo *devtmpfs* utilizado inicialmente.

9.3.2.3. Carga de módulos

Los controladores de dispositivo compilados como módulos pueden tener alias integrados. Los alias son visibles en la salida del programa **modinfo** y suelen estar relacionados con los identificadores específicos del bus de los dispositivos compatibles con un módulo. Por ejemplo, el controlador *snd-fm801* admite dispositivos PCI con ID de proveedor 0x1319 e ID de dispositivo 0x0801, y tiene el alias *pci:v00001319d00000801sv*sd*bc04sc01i**. Para la mayoría de los dispositivos, el controlador de bus exporta el alias del controlador que gestionaría el dispositivo mediante *sysfs*. Por ejemplo, el archivo */sys/bus/pci/devices/0000:00:0d.0/modalias* podría contener la cadena *pci:v00001319d00000801sv00001319sd00001319bc04sc01i00*. Las reglas predeterminadas de udev harán que **udevd** llame a **/sbin/modprobe** con el contenido de la variable de entorno uevent *MODALIAS* (que debería ser el mismo que el del archivo modalias en sysfs), cargando así todos los módulos cuyos alias coincidan con esta cadena después de la expansión con comodines.

En este ejemplo, esto significa que, además de *snd-fm801*, se cargará el controlador *forte* obsoleto (e indeseado) si está disponible. Consulte a continuación cómo evitar la carga de controladores indeseado.

El propio kernel también puede cargar módulos para protocolos de red, sistemas de archivos y compatibilidad con NLS bajo demanda.

9.3.2.4. Manejo de dispositivos conectables en caliente/dinámicos

Al conectar un dispositivo, como un reproductor de MP3 USB (Universal Serial Bus), el kernel reconoce que el dispositivo está conectado y genera un evento UE. Este evento UE es gestionado por **udevd** como se describió anteriormente.

9.3.3. Problemas con la carga de módulos y la creación de dispositivos

Existen algunos problemas posibles al crear automáticamente nodos de dispositivos.

9.3.3.1. Un módulo del kernel no se carga automáticamente

Udev solo cargará un módulo si tiene un alias específico del bus y el controlador del bus exporta correctamente los alias necesarios a *sysfs*. En otros casos, se debe gestionar la carga del módulo por otros medios. Con Linux-6.13.4, udev carga controladores correctamente escritos para dispositivos INPUT, IDE, PCI, USB, SCSI, SERIO y FireWire.

Para determinar si el controlador de dispositivo que necesita es compatible con udev, ejecute **modinfo** con el nombre del módulo como argumento. Ahora, busque el directorio del dispositivo en `/sys/bus` y compruebe si contiene el archivo `modalias`.

Si el archivo `modalias` existe en `sysfs`, el controlador es compatible con el dispositivo y puede comunicarse con él directamente, pero no tiene el alias; se trata de un error del controlador. Cargue el controlador sin la ayuda de udev y espere a que el problema se solucione más adelante.

Si no hay ningún archivo `modalias` en el directorio correspondiente en `/sys/bus`, significa que los desarrolladores del kernel aún no han añadido compatibilidad con modalias para este tipo de bus. Con Linux-6.13.4, esto ocurre con los buses ISA. Se espera que este problema se solucione en versiones posteriores del kernel.

Udev no está diseñado para cargar controladores "wrapper" como *snd-pcm-oss* ni controladores no hardware como *loop*.

9.3.3.2. Un módulo del kernel no se carga automáticamente y Udev no está diseñado para cargarlo.

Si el módulo "wrapper" solo mejora la funcionalidad de otro módulo (por ejemplo, *snd-pcm-oss* mejora la funcionalidad de *snd-pcm* al permitir que las tarjetas de sonido estén disponibles para aplicaciones OSS), configure **modprobe** para que cargue el wrapper después de que udev cargue el módulo encapsulado. Para ello, añada una línea "softdep" al archivo `/etc/modprobe.d/<filename>.conf` correspondiente. Por ejemplo:

```
softdep snd-pcm post: snd-pcm-oss
```

Tenga en cuenta que el comando "softdep" también permite dependencias `pre:` o una combinación de ambas dependencias `pre:` y `post:`.

Consulte la página del manual de *modprobe.d(5)* para obtener más información sobre la sintaxis y las capacidades de "softdep".

Si el módulo en cuestión no es un wrapper y es útil por sí solo, configure el script de arranque de módulos para que cargue este módulo al arrancar el sistema. Para ello, añada el nombre del módulo al archivo `/etc/sysconfig/modules` en una línea aparte. Esto también funciona con módulos wrapper, pero no es óptimo en ese caso.

9.3.3.3. Udev carga un módulo no deseado

No compile el módulo o agréguelo a la lista negra en el archivo */etc/modprobe.d/blacklist.conf*, como se hizo con el módulo forte en el ejemplo siguiente:

```
blacklist forte
```

Los módulos incluidos en la lista negra se pueden cargar manualmente con el comando explícito **modprobe**.

9.3.3.4. Udev crea un dispositivo incorrectamente o crea un enlace simbólico erróneo.

Esto suele ocurrir si una regla coincide inesperadamente con un dispositivo. Por ejemplo, una regla mal escrita puede coincidir tanto con un disco SCSI (como se desea) como con el dispositivo genérico SCSI correspondiente (incorrectamente) por proveedor. Encuentre la regla problemática y especifíquela con el comando **udevadm info.**

9.3.3.5. La regla de Udev funciona de forma poco fiable.

Esta puede ser otra manifestación del problema anterior. Si no es así, y su regla utiliza atributos `sysfs`, podría tratarse de un problema de sincronización del kernel, que se solucionará en kernels posteriores. Por ahora, puede solucionarlo creando una regla que espere el atributo `sysfs` utilizado y agregándola al archivo `/etc/udev/rules.d/10-wait_for_sysfs.rules` (cree este archivo si no existe). Por favor, notifique a la lista de desarrollo de LFS si lo hace y le resulta útil.

9.3.3.6. Udev no crea un dispositivo

Primero, asegúrese de que el controlador esté integrado en el kernel o ya cargado como módulo, y de que udev no esté creando un dispositivo con un nombre incorrecto.

Si un controlador del kernel no exporta sus datos a *sysfs*, udev carece de la información necesaria para crear un nodo de dispositivo. Esto es más probable que ocurra con controladores de terceros externos al árbol del kernel. Cree un nodo de dispositivo estático en `/usr/lib/udev/devices` con los números de dispositivo mayor/menor adecuados (consulte el archivo `devices.txt` en la documentación del kernel o la documentación proporcionada por el proveedor del controlador externo). **udev** copiará el nodo de dispositivo estático a `/dev`.

9.3.3.7. El orden de nombres de los dispositivos cambia aleatoriamente después de reiniciar

Esto se debe a que udev, por diseño, gestiona los eventos y carga los módulos en paralelo y, por lo tanto, en un orden impredecible. Esto nunca se solucionará. No debe confiar en la estabilidad de los nombres de los dispositivos del kernel. En su lugar, cree sus propias reglas que creen enlaces simbólicos con nombres estables basándose en algunos atributos estables del dispositivo, como un número de serie o la salida de varias utilidades *_id instaladas por udev. Consulte la Sección 9.4, "Administración de dispositivos" y la Sección 9.5, "Configuración general de red" para ver ejemplos.

9.3.4. Lecturas útiles

Puede encontrar documentación adicional útil en los siguientes sitios:

• Implementación de *devfs* en el espacio de usuario:

http://www.kroah.com/linux/talks/ols_2003_udev_paper/Reprint-Kroah-Hartman-OLS2003.pdf

• El sistema de archivos *sysfs*:

https://www.kernel.org/pub/linux/kernel/people/mochel/doc/papers/ols-2005/mochel.pdf

9.4. Administración de dispositivos

9.4.1. Dispositivos de red

Udev, por defecto, nombra los dispositivos de red según los datos del firmware/BIOS o características físicas como el bus, la ranura o la dirección MAC. El propósito de esta convención de nomenclatura es garantizar que los dispositivos de red tengan nombres coherentes, no según el momento en que se detectó la tarjeta de red. En versiones anteriores de Linux (por ejemplo, en un ordenador con dos tarjetas de red de Intel y Realtek), la tarjeta de red de Intel podría haberse convertido en eth0, mientras que la de Realtek en eth1. Tras reiniciar, a veces las tarjetas se renumeraban al revés.

En el nuevo esquema de nomenclatura, los nombres típicos de los dispositivos de red son similares a enp5s0 o wlp3s0. Si no se desea esta convención de nomenclatura, se puede implementar el esquema de nomenclatura tradicional o uno personalizado.

9.4.1.1. Desactivación de la nomenclatura persistente en la línea de comandos del kernel

El esquema de nomenclatura tradicional, con eth0, eth1, etc., se puede restaurar añadiendo **net.ifnames=0** en la línea de comandos del kernel. Esto es más adecuado para sistemas con un solo dispositivo Ethernet de un tipo específico. Los portátiles suelen tener dos conexiones Ethernet llamadas eth0 y wlan0; estos portátiles también pueden usar este método. La línea de comandos se encuentra en el archivo de configuración de GRUB. Consulte la Sección 10.4.4, “Creación del archivo de configuración de GRUB”.

9.4.1.2. Creación de reglas udev personalizadas

El esquema de nomenclatura se puede personalizar creando reglas udev personalizadas. Se ha incluido un script que genera las reglas iniciales. Genere estas reglas ejecutando:

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

Ahora, inspeccione el archivo */etc/udev/rules.d/70-persistent-net.rules* para averiguar qué nombre se asignó a cada dispositivo de red:

```
cat /etc/udev/rules.d/70-persistent-net.rules
```

Nota

En algunos casos, como cuando se asignan direcciones MAC manualmente a una tarjeta de red o en un entorno virtual como Qemu o Xen, es posible que el archivo de reglas de red no se genere porque las direcciones no se asignan de forma consistente. En estos casos, este método no se puede utilizar.

El archivo comienza con un bloque de comentarios, seguido de dos líneas para cada NIC. La primera línea de cada NIC es una descripción comentada que muestra sus identificadores de hardware (por ejemplo, su proveedor PCI y los identificadores del dispositivo, si se trata de una tarjeta PCI), junto con

su controlador (entre paréntesis, si se encuentra). Ni el ID del hardware ni el controlador se utilizan para determinar el nombre de una interfaz; esta información es solo de referencia. La segunda línea corresponde a la regla udev que coincide con esta NIC y le asigna un nombre.

Todas las reglas udev se componen de varias palabras clave, separadas por comas y espacios opcionales. A continuación, se muestran las palabras clave y una explicación de cada una:

• `SUBSYSTEM=="net"`: Indica a udev que ignore los dispositivos que no sean tarjetas de red.

• `ACTION=="add"`: Indica a udev que ignore esta regla para un uevent que no sea una adición (los uevents "eliminar" y "cambiar" también ocurren, pero no es necesario cambiar el nombre de las interfaces de red).

• `DRIVERS=="?*"` - Esto existe para que udev ignore las subinterfaces VLAN o de puente (ya que estas subinterfaces no tienen controladores). Estas subinterfaces se omiten porque el nombre que se les asignaría entraría en conflicto con los dispositivos principales.

• `ATTR{address}` - El valor de esta palabra clave es la dirección MAC de la NIC.

• `ATTR{type}=="1"` - Esto garantiza que la regla solo coincida con la interfaz principal en el caso de ciertos controladores inalámbricos que crean múltiples interfaces virtuales. Las interfaces secundarias se omiten por la misma razón que las subinterfaces VLAN y de puente: de lo contrario, se produciría un conflicto de nombres.

• `NAME` - El valor de esta palabra clave es el nombre que udev asignará a esta interfaz.

El valor de *NAME* es la parte importante. Asegúrese de saber qué nombre se ha asignado a cada una de sus tarjetas de red antes de continuar y asegúrese de usar ese valor de NAME al crear sus archivos de configuración de red. Incluso si se crea el archivo de regla udev personalizado, udev puede asignar uno o más nombres alternativos a una NIC según sus características físicas. Si una regla udev personalizada renombra una NIC usando un nombre ya asignado como nombre alternativo de otra NIC, esta regla fallará. Si esto ocurre, puede crear el archivo de configuración de enlace */etc/udev/network/99-default*.

con una política de asignación alternativa vacía, anulando así el archivo de configuración predeterminado */usr/lib/udev/network/99-default.link*:

```
sed -e '/^AlternativeNamesPolicy/s/=.*$/=/'   \
       /usr/lib/udev/network/99-default.link \
     > /etc/udev/network/99-default.link
```

9.4.2. Enlaces simbólicos de CD-ROM

Algunos programas que desee instalar posteriormente (por ejemplo, varios reproductores multimedia) requieren que los enlaces simbólicos */dev/cdrom* y */dev/dvd* existan y apunten a un dispositivo de CD-ROM o DVD-ROM. Además, puede ser conveniente incluir referencias a estos enlaces simbólicos en */etc/fstab*. Udev incluye un script que genera archivos de reglas para crear estos enlaces simbólicos, según las capacidades de cada dispositivo, pero debe decidir cuál de los dos modos de operación desea que utilice el script.

En primer lugar, el script puede operar en modo "por ruta" (predeterminado para dispositivos USB y FireWire), donde las reglas que crea dependen de la ruta física al dispositivo de CD o DVD. En segundo lugar, puede operar en modo "por ID" (predeterminado para dispositivos IDE y SCSI), donde las reglas que crea dependen de las cadenas de identificación almacenadas en el propio dispositivo de CD o DVD. La ruta se determina mediante el script **path_id** de udev, y las cadenas de identificación se leen desde el hardware mediante sus programas **ata_id** o **scsi_id**, según el tipo de dispositivo.

Cada enfoque tiene sus ventajas; el correcto depende del tipo de cambios que puedan ocurrir en el dispositivo.

Si prevé que la ruta física del dispositivo (es decir, los puertos o ranuras donde se conecta) cambie, por ejemplo, porque planea mover la unidad a un puerto IDE diferente o a un conector USB diferente, debería usar el modo "por id". Por otro lado, si prevé que la identificación del dispositivo cambie, por ejemplo, porque podría dejar de funcionar y pretende reemplazarlo por otro dispositivo que se conecte a los mismos conectores, debería usar el modo "por ruta".

Si su unidad admite cualquiera de los dos tipos de cambios, elija un modo según el tipo de cambio que prevea que ocurra con mayor frecuencia.

Importante

Los dispositivos externos (por ejemplo, una unidad de CD conectada por USB) no deben usar la persistencia por ruta, ya que cada vez que se conectan a un nuevo puerto externo, su ruta física cambia. Todos los dispositivos conectados externamente tendrán este problema si se escriben reglas de udev para reconocerlos por su ruta física; el problema no se limita a las unidades de CD y DVD.

Si desea ver los valores que usarán los scripts de udev, para el dispositivo de CD-ROM correspondiente, busque el directorio correspondiente en */sys* (por ejemplo, */sys/block/hdd*) y ejecute un comando similar al siguiente:

```
udevadm test /sys/block/hdd
```

Observe las líneas que contienen la salida de varios programas *_id. El modo "by-id" usará el valor ID_SERIAL si existe y no está vacío; de lo contrario, usará una combinación de ID_MODEL e ID_REVISION. El modo "by-path" usará el valor ID_PATH. Si el modo predeterminado no se adapta a su situación, puede realizar la siguiente modificación en el archivo */etc/udev/rules.d/83-cdrom-symlinks.rules*, como se indica a continuación (donde mode es "by-id" o "by-path"):

```
sed -e 's/"write_cd_rules"/"write_cd_rules mode"/' \
    -i /etc/udev/rules.d/83-cdrom-symlinks.rules
```

Tenga en cuenta que no es necesario crear los archivos de reglas ni los enlaces simbólicos en este momento, ya que ha montado mediante enlace el directorio `/dev` del host en el sistema LFS y asumimos que los enlaces simbólicos existen en el host. Las reglas y los enlaces simbólicos se crearán la primera vez que inicie el sistema LFS.

Sin embargo, si tiene varios dispositivos de CD-ROM, los enlaces simbólicos generados en ese momento podrían apuntar a dispositivos diferentes a los que apuntan en su host, ya que los dispositivos

no se detectan en un orden predecible. Las asignaciones creadas al iniciar el sistema LFS por primera vez serán estables, por lo que esto solo supone un problema si necesita que los enlaces simbólicos de ambos sistemas apunten al mismo dispositivo. Si es necesario, inspeccione (y posiblemente edite) el archivo `/etc/udev/rules.d/70-persistent-cd.rules` generado tras el arranque para asegurarse de que los enlaces simbólicos asignados se ajusten a sus necesidades.

9.4.3. Gestión de dispositivos duplicados

Como se explica en la Sección 9.3, “Descripción general del manejo de dispositivos y módulos”, el orden en que aparecen los dispositivos con la misma función en */dev* es prácticamente aleatorio. Por ejemplo, si tiene una cámara web USB y un sintonizador de TV, a veces */dev/video0* se refiere a la cámara y */dev/video1* al sintonizador; en ocasiones, tras reiniciar, el orden cambia. Para todo tipo de hardware, excepto tarjetas de sonido y de red, esto se puede solucionar creando reglas udev para crear enlaces simbólicos persistentes. El caso de las tarjetas de red se trata por separado en la Sección 9.5, “Configuración general de red”, y la configuración de la tarjeta de sonido se puede encontrar en *BLFS*.

Para cada dispositivo que pueda tener este problema (incluso si no existe en su distribución de Linux actual), busque el directorio correspondiente en */sys/class* o */sys/block*. Para dispositivos de vídeo, este directorio puede ser */sys/class/video4linux/videoX*. Determine los atributos que identifican el dispositivo de forma única (normalmente, los ID de proveedor y producto o los números de serie funcionan):

```
udevadm info -a -p /sys/class/video4linux/video0
```

Luego, escriba las reglas que creen los enlaces simbólicos, por ejemplo:

```
cat > /etc/udev/rules.d/83-duplicate_devs.rules << "EOF"

# Enlaces simbólicos persistentes para la cámara web y el sintonizador
KERNEL=="video*", ATTRS{idProduct}=="1910", ATTRS{idVendor}=="0d81", SYMLINK+="webcam"
KERNEL=="video*", ATTRS{device}=="0x036f",  ATTRS{vendor}=="0x109e", SYMLINK+="tvtuner"

EOF
```

El resultado es que los dispositivos */dev/video0* y */dev/video1* siguen haciendo referencia aleatoriamente a sintonizador y la cámara web (y por lo tanto nunca deben usarse directamente), pero hay enlaces simbólicos */dev/tvtuner* y */dev/webcam* que siempre apuntan al dispositivo correcto.

9.5. Configuración general de red

9.5.1. Creación de archivos de configuración de interfaz de red

Los archivos en */etc/sysconfig/* suelen determinar qué interfaces se activan y desactivan mediante el script de red. Este directorio debe contener un archivo para cada interfaz a configurar, como *ifconfig.xyz*, donde "xyz" describe la tarjeta de red. El nombre de la interfaz (p. ej., eth0) suele ser adecuado. Cada archivo contiene los atributos de una interfaz, como su(s) dirección(es) IP, máscaras de subred, etc. La raíz del nombre del archivo debe ser *ifconfig*.

Nota

Si no se utilizó el procedimiento de la sección anterior, udev asignará nombres de interfaz a la tarjeta de red según las características físicas del sistema, como enp2s1. Si no está seguro del nombre de su interfaz, puede ejecutar **ip link** o **ls /sys/class/net** después de iniciar el sistema.

Los nombres de las interfaces dependen de la implementación y configuración del demonio udev que se ejecuta en el sistema. El demonio udev para LFS (instalado en la Sección 8.76, “Udev desde Systemd-257.3”) no se ejecutará hasta que se inicie el sistema LFS. Por lo tanto, los nombres de las interfaces en el sistema LFS no siempre se pueden determinar ejecutando estos comandos en la distribución del host, *ni siquiera en el entorno chroot*.

El siguiente comando crea un archivo de ejemplo para el dispositivo eth0 con una dirección IP estática:

```
cd /etc/sysconfig/
cat > ifconfig.eth0 << "EOF"
ONBOOT=yes
IFACE=eth0
SERVICE=ipv4-static
IP=192.168.1.2
GATEWAY=192.168.1.1
PREFIX=24
BROADCAST=192.168.1.255
EOF
```

Los valores en cursiva deben modificarse en cada archivo para configurar las interfaces correctamente.

Si la variable *ONBOOT* se establece en *sí*, el script de red de System V activará la tarjeta de interfaz de red (NIC) durante el proceso de arranque del sistema. Si se establece en cualquier valor distinto de *sí*, el script de red ignorará la NIC y no se iniciará automáticamente. Las interfaces se pueden iniciar o detener manualmente con los comandos **ifup** e **ifdown**.

La variable *IFACE* define el nombre de la interfaz, por ejemplo, eth0. Es necesaria para todos los archivos de configuración de dispositivos de red. La extensión del nombre de archivo debe coincidir con este valor.

La variable *SERVICE* define el método utilizado para obtener la dirección IP. El paquete LFS-Bootscripts tiene un formato modular de asignación de IP, y la creación de archivos adicionales en el directorio */lib/services/* permite otros métodos de asignación de IP.

Esto se utiliza comúnmente para el Protocolo de Configuración Dinámica de Host (DHCP), que se aborda en el manual de BLFS.

La variable *GATEWAY* debe contener la dirección IP de la puerta de enlace predeterminada, si existe. De lo contrario, comente la variable por completo.

La variable *PREFIX* especifica el número de bits utilizados en la subred. Cada segmento de una dirección IP tiene 8 bits. Si la máscara de red de la subred es 255.255.255.0, se utilizan los tres primeros segmentos (24 bits) para especificar el número de red. Si la máscara de red es 255.255.255.240, la subred utiliza los primeros 28 bits. Los proveedores de servicios de Internet (ISP) DSL y por cable suelen utilizar prefijos de más de 24 bits. En este ejemplo (PREFIX=24), la máscara de red es 255.255.255.0. Ajuste la variable PREFIX según su subred específica. Si se omite, el valor predeterminado de PREFIX es 24.

Para obtener más información, consulte la página de manual de **ifup**.

9.5.2. Creación del archivo /etc/resolv.conf

El sistema necesitará algún método para obtener la resolución de nombres del Servicio de Nombres de Dominio (DNS) para convertir los nombres de dominio de Internet en direcciones IP y viceversa. La mejor manera de lograrlo es introducir la dirección IP del servidor DNS, disponible a través del ISP o del administrador de red, en */etc/resolv.conf*. Cree el archivo ejecutando lo siguiente:

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

domain <Su nombre de dominio>
nameserver <Dirección IP de su servidor de nombres principal>
nameserver <Dirección IP de su servidor de nombres secundario>

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

La instrucción *domain* puede omitirse o reemplazarse con una instrucción de búsqueda. Consulte la página del manual de *resolv.conf* para obtener más detalles.

Reemplace *<Dirección IP del servidor de nombres>* con la dirección IP del DNS más adecuada para la configuración. A menudo habrá más de una entrada (los requisitos exigen servidores secundarios como respaldo). Si solo necesita o desea un servidor DNS, elimine la segunda línea del *servidor de nombres* del archivo. La dirección IP también puede ser la de un enrutador en la red local.

Nota

Las direcciones DNS IPv4 públicas de Google son 8.8.8.8 y 8.8.4.4.

9.5.3. Configuración del nombre de host del sistema

Durante el proceso de arranque, el archivo */etc/hostname* se utiliza para establecer el nombre de host del sistema.

Cree el archivo */etc/hostname* e introduzca un nombre de host ejecutando:

```
echo "<lfs>" > /etc/hostname
```

<lfs> debe reemplazarse por el nombre asignado al equipo. No introduzca aquí el nombre de dominio completo (FQDN). Esa información se guarda en el archivo */etc/hosts*.

9.5.4. Personalización del archivo /etc/hosts

Decida un nombre de dominio completo (FQDN) y posibles alias para usar en el archivo */etc/hosts*. Si usa direcciones IP estáticas, también deberá elegir una dirección IP. La sintaxis para una entrada en el archivo hosts es:

```
IP_address myhost.example.org alias
```

A menos que el equipo vaya a ser visible en Internet (es decir, que exista un dominio registrado y un bloque válido de direcciones IP asignadas; la mayoría de los usuarios no lo tienen), asegúrese de que la dirección IP esté dentro del rango de direcciones IP de la red privada. Los rangos válidos son:

```
Rango de direcciones de red privada Prefijo normal
10.0.0.1 - 10.255.255.254            8
172.x.0.1 - 172.x.255.254           16
192.168.y.1 - 192.168.y.254         24
```

"x" puede ser cualquier número entre 16 y 31. "y" puede ser cualquier número entre 0 y 255.

Una dirección IP privada válida podría ser 192.168.1.2.

Para que el ordenador sea visible en Internet, un FQDN válido puede ser el propio nombre de dominio o una cadena resultante de la concatenación de un prefijo (generalmente el nombre de host) y el nombre de dominio con un carácter "." Además, debe contactar al proveedor del dominio para resolver el FQDN a su dirección IP pública.

Incluso si el ordenador no es visible en Internet, un FQDN sigue siendo necesario para que ciertos programas, como los MTA, funcionen correctamente. Para ello, se puede utilizar un FQDN especial, *localhost.localdomain*.

Cree el archivo */etc/hosts* ejecutando:

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

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <FQDN> <HOSTNAME>
<192.168.1.2> <FQDN> <HOSTNAME> [alias1] [alias2 ...]
::1       localhost ip6-localhost ip6-loopback
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters

# End /etc/hosts
EOF
```

Los valores *<192.168.1.2>*, *<FQDN>* y *<HOSTNAME>* deben modificarse para usos o requisitos específicos (si un administrador de red/sistema le asigna una dirección IP y el equipo se conectará a una red existente). Los alias opcionales pueden omitirse.

9.6. Uso y configuración de los scripts de arranque de System V

9.6.1. ¿Cómo funcionan los scripts de arranque de System V?

Esta versión de LFS utiliza una herramienta de arranque especial llamada SysVinit, basada en una serie de niveles de ejecución. El procedimiento de arranque puede variar considerablemente de un sistema a otro; el hecho de que las cosas funcionen de una manera en una distribución de Linux en particular no garantiza que funcionen de la misma manera en LFS. LFS tiene su propia forma de hacer las cosas, pero respeta los estándares generalmente aceptados.

Existe un procedimiento de arranque alternativo llamado **systemd**. No profundizaremos en este proceso de arranque. Para una descripción detallada, visite:

https://www.linux.com/training-tutorials/understanding-and-using-systemd/.

SysVinit (al que nos referiremos como "init" de ahora en adelante) utiliza un esquema de niveles de ejecución. Hay siete niveles de ejecución, numerados del 0 al 6. (En realidad, hay más niveles de ejecución, pero los demás son para casos especiales y generalmente no se usan. Consulte init(8) para más detalles). Cada uno de los siete corresponde a las acciones que el equipo debe realizar al iniciarse o apagarse. El nivel de ejecución predeterminado es 3. A continuación, se detallan los diferentes niveles de ejecución tal como se implementan en LFS:

```
0: detener el equipo
1: modo monousuario
2: reservado para personalización; en caso contrario, igual que 3
3: modo multiusuario con conexión en red
4: reservado para personalización; en caso contrario, igual que 3
5: igual que 4, se suele usar para iniciar sesión mediante interfaz gráfica (como
gdm de GNOME o lxdm de LXDE)
6: reiniciar el equipo
```

Nota

Clásicamente, el nivel de ejecución 2 anterior se definía como "modo multiusuario sin conexión de red", pero esto solo ocurría hace muchos años, cuando varios usuarios podían conectarse a un sistema mediante puertos serie. En el entorno actual, esto no tiene sentido, y ahora decimos que está "reservado".

9.6.2. Configuración de SysVinit

Durante la inicialización del núcleo, el primer programa que se ejecuta (si no se sobrescribe en la línea de comandos) es **init**. Este programa lee el archivo de inicialización `/etc/inittab`. Cree este archivo con:

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

id:3:initdefault:

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

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

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

su:S06:once:/sbin/sulogin
s1:1:respawn:/sbin/sulogin

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

# End de /etc/inittab
EOF
```

Una explicación de esto El archivo de inicialización se encuentra en la página del manual de inittab. En LFS, el comando clave es **rc**. El archivo de inicialización anterior indica a **rc** que ejecute todos los scripts que empiezan con una S en el directorio */etc/rc.d/rcS.d*, seguidos de todos los scripts que empiezan con una S en el directorio */etc/rc.d/rc?.d* donde el signo de interrogación se especifica mediante el valor predeterminado de init.

Para mayor comodidad, el script **rc** lee una biblioteca de funciones en */lib/lsb/init-functions*. Esta biblioteca también lee un archivo de configuración opcional, */etc/sysconfig/rc.site*. Cualquiera de los parámetros de configuración del sistema descritos en las secciones posteriores se puede colocar en este archivo, lo que permite consolidar todos los parámetros del sistema en un solo archivo.

Para mayor comodidad durante la depuración, el script functions también registra toda la salida en */run/var/bootlog*. Dado que el directorio */run* es un archivo tmpfs, este archivo no es persistente entre arranques; Sin embargo, se añade al archivo más permanente */var/log/boot.log* al final del proceso de arranque.

9.6.2.1. Cambio de niveles de ejecución

El cambio de niveles de ejecución se realiza con **init <*runlevel*>**, donde <*runlevel*> es el nivel de ejecución objetivo. Por ejemplo, para reiniciar el equipo, un usuario podría ejecutar el comando i**nit 6**, que es un alias del comando **reboot**. Asimismo, **init 0** es un alias del comando **halt**.

Hay varios directorios en */etc/rc.d* que se parecen a *rc?.d* (donde ? es el número del nivel de ejecución) y *rcS.d*, todos con enlaces simbólicos. Algunos enlaces empiezan con K; otros, con S, y todos tienen dos números después de la letra inicial. La K significa detener (matar) un servicio y la S significa iniciarlo. Los números determinan el orden de ejecución de los scripts, del 00 al 99; cuanto

menor sea el número, antes se ejecutará el script. Cuando **init** cambia a otro nivel de ejecución, los servicios correspondientes se inician o detienen, según el nivel de ejecución seleccionado.

Los scripts reales se encuentran en */etc/rc.d/init.d*. Realizan el trabajo real, y todos los enlaces simbólicos apuntan a ellos. Los enlaces K y S apuntan al mismo script en */etc/rc.d/init.d*. Esto se debe a que los scripts pueden llamarse con diferentes parámetros, como *inicio (start)*, *parada (stop)*, *reinicio (restart)*, *recarga (reload)* y *estado (status)*. Cuando se encuentra un enlace K, se ejecuta el script correspondiente con el argumento de parada. Cuando se encuentra un enlace S, se ejecuta el script correspondiente con el argumento de `inicio`.

Estas son descripciones de lo que los argumentos hacen que hagan los scripts:

inicio (start)

Se inicia el servicio.

parada (stop)

Se detiene el servicio.

reinicio (restart)

Se detiene el servicio y se vuelve a iniciar.

recarga (reload)

Se actualiza la configuración del servicio. Esto se utiliza después de modificar el archivo de configuración de un servicio, cuando no es necesario reiniciarlo.

estado (status)

Indica si el servicio se está ejecutando y con qué PID.

Siéntase libre de modificar el funcionamiento del proceso de arranque (después de todo, es su propio sistema LFS). Los archivos que se proporcionan aquí son un ejemplo de cómo se puede hacer.

9.6.3. Scripts de arranque de Udev

El script de inicio */etc/rc.d/init.d/udev* inicia **udevd**, activa cualquier dispositivo "coldplug" ya creado por el kernel y espera a que se completen las reglas. El script también desactiva el controlador de uevents del valor predeterminado de */sbin/hotplug*. Esto se debe a que el kernel ya no necesita llamar a un binario externo. En su lugar, **udevd** escuchará en un socket netlink los uevents que el kernel genere.

El script **/etc/rc.d/init.d/udev_retry** se encarga de reactivar eventos para subsistemas cuyas reglas puedan depender de sistemas de archivos que no estén montados hasta que se ejecute el script **mountfs** (en particular, */usr* y */var* pueden causar esto). Este script se ejecuta después del script **mountfs**, por lo que esas reglas (si se reactivan) deberían funcionar correctamente la segunda vez. Se configura mediante el archivo */etc/sysconfig/udev_retry*. Cualquier palabra en este archivo, excepto comentarios, se considera un nombre de subsistema que se activará al reintentar. Para encontrar el

subsistema de un dispositivo, utilice **udevadm info --attribute-walk <device>**, donde <*device*> es una ruta absoluta en /dev o /sys, como */dev/sr0* o */sys/class/rtc*.

Para obtener información sobre la carga de módulos del kernel y udev, consulte la Sección 9.3.2.3, “Carga de módulos”.

9.6.4. Configuración del reloj del sistema

El script **setclock** lee la hora del reloj del hardware, también conocido como reloj de la BIOS o del Semiconductor de Óxido Metálico Complementario (CMOS). Si el reloj del hardware está configurado en UTC, este script convertirá la hora del reloj del hardware a la hora local mediante el archivo */etc/localtime* (que indica al programa **hwclock** la zona horaria que debe usar). No hay forma de detectar si el reloj del hardware está configurado en UTC, por lo que debe configurarse manualmente.

El programa **setclock** se ejecuta mediante udev cuando el kernel detecta la capacidad del hardware al arrancar. También se puede ejecutar manualmente con el parámetro "stop" para almacenar la hora del sistema en el reloj CMOS.

Si no recuerda si el reloj del hardware está configurado en UTC, averígüelo ejecutando el comando "**hwclock --localtime--show**". Esto mostrará la hora actual según el reloj del hardware. Si esta hora coincide con la que indica su reloj, el reloj del hardware está configurado en hora local. Si la salida de "**hwclock**" no es la hora local, es probable que esté configurado en UTC. Verifíquelo sumando o restando las horas correspondientes a su zona horaria a la hora que muestra "**hwclock**". Por ejemplo, si se encuentra en la zona horaria MST, también conocida como GMT -0700, sume siete horas a la hora local.

Cambie el valor de la variable *UTC* a `0` (cero) si el reloj del hardware *NO* está configurado en hora UTC.

Cree un nuevo archivo */etc/sysconfig/clock* ejecutando lo siguiente:

```
cat > /etc/sysconfig/clock << "EOF"
# Begin /etc/sysconfig/clock

UTC=1

# Configure aquí las opciones que necesite para hwclock,
# como el tipo de reloj de hardware de la máquina para Alphas.
CLOCKPARAMS=

# End /etc/sysconfig/clock
EOF
```

En https://www.linuxfromscratch.org/hints/downloads/files/time.txt se encuentra una buena sugerencia sobre cómo gestionar la hora en LFS. Explica aspectos como las zonas horarias, la hora UTC y la variable de entorno TZ.

Nota

Los parámetros CLOCKPARAMS y UTC también se pueden configurar en el archivo */etc/sysconfig/rc.site*.

9.6.5. Configuración de la consola de Linux

Esta sección explica cómo configurar el script de arranque de la consola que configura la asignación de teclado, la fuente de la consola y el nivel de registro del kernel de la consola. Si no se van a utilizar caracteres no ASCII (por ejemplo, el símbolo de copyright, la libra esterlina y el símbolo del euro) y el teclado es estadounidense, se puede omitir gran parte de esta sección. Sin el archivo de configuración (o la configuración equivalente en `rc.site`), el script de arranque de la consola no hará nada.

El script de consola lee el archivo `/etc/sysconfig/console` para obtener información de configuración. Decida qué mapa de teclas y fuente de pantalla se utilizarán. Diversos tutoriales específicos para cada idioma también pueden ser útiles; consulte https://tldp.org/HOWTO/HOWTO-INDEX/other-lang.html. Si aún tiene dudas, consulte los directorios `/usr/share/keymaps` y `/usr/share/consolefonts` para encontrar mapas de teclas y fuentes de pantalla válidos. Lea las páginas del manual l*oadkeys(1)* y *setfont(8)* para determinar los argumentos correctos para estos programas.

El archivo `/etc/sysconfig/console` debe contener líneas con el formato: `VARIABLE=valor`. Se reconocen las siguientes variables:

LOGLEVEL

> Esta variable especifica el nivel de registro de los mensajes del kernel enviados a la consola, según lo establecido por **dmesg -n**. Los niveles válidos van del 1 (sin mensajes) al 8. El nivel predeterminado es 7, que es bastante detallado.

KEYMAP

> Esta variable especifica los argumentos del programa **loadkeys**; normalmente, el nombre del mapa de teclas que se cargará, por ejemplo, *it*. Si esta variable no está establecida, el script de arranque no ejecutará el programa **loadkeys** y se utilizará el mapa de teclas predeterminado del kernel. Tenga en cuenta que algunos mapas de teclas tienen varias versiones con el mismo nombre (cz y sus variantes en qwerty/ y qwertz/, es en olpc/ y qwerty/, y trf en fgGIod/ y qwerty/). En estos casos, también se debe especificar el directorio principal (por ejemplo, qwerty/es) para garantizar que se cargue el mapa de teclas adecuado.

KEYMAP_CORRECTIONS

> Esta variable (poco utilizada) especifica los argumentos para la segunda llamada al programa **loadkeys**. Resulta útil si el mapa de teclas predeterminado no es del todo satisfactorio y es necesario realizar un pequeño ajuste. Por ejemplo, para incluir el símbolo del euro en un mapa de teclas que normalmente no lo tiene, establezca esta variable en *euro2*.

FFONT

> Esta variable especifica los argumentos para el programa **setfont**. Normalmente, incluye el nombre de la fuente, -m, y el nombre del mapa de caracteres de la aplicación que se va a cargar. Por ejemplo, para cargar la fuente "lat1-16" junto con el mapa de caracteres de la aplicación "8859-1" (apropiado en EE. UU.), establezca esta variable en *lat1-16 -m 8859-1*. En modo UTF-8, el núcleo utiliza el mapa de caracteres de la aplicación para convertir códigos de teclas

de 8 bits a UTF-8. Por lo tanto, el argumento del parámetro "-m" debe establecerse en la codificación de los códigos de teclas compuestos en el mapa de teclas.

UNICODE

Establezca esta variable en 1, sí o verdadero para que la consola esté en modo UTF-8. Esto es útil en configuraciones regionales basadas en UTF-8 y perjudicial en caso contrario.

LEGACY_CHARSET

Para muchas distribuciones de teclado, no existe un mapa de teclas Unicode estándar en el paquete Kbd. El script de arranque de la **consola** convertirá un mapa de teclas disponible a UTF-8 sobre la marcha si esta variable se establece en la codificación del mapa de teclas disponible que no sea UTF-8.

Algunos ejemplos:

• Usaremos C.UTF-8 como configuración regional para las sesiones interactivas en la consola de Linux en la Sección 9.7, “Configuración de la configuración regional del sistema”, por lo que debemos establecer UNICODE en 1. Las fuentes de consola incluidas en el paquete Kbd, que contienen los glifos de todos los caracteres de los mensajes del programa en la configuración regional *C.UTF-8*, son *LatArCyrHeb*.psfu.gz*, *LatGrkCyr*.psfu.gz*, *Lat2-Terminus16.psfu.gz* y *pancyrillic.f16.psfu.gz* en */usr/share/consolefonts* (las otras fuentes de consola incluidas carecen de los glifos de algunos caracteres, como las comillas Unicode izquierdas/derechas y el guion Unicode inglés). Por lo tanto, establezca una de ellas, por ejemplo, *Lat2-Terminus16.psfu.gz*, como la fuente de consola predeterminada.

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
FONT="Lat2-Terminus16"

# End /etc/sysconfig/console
EOF
```

• Para una configuración no Unicode, generalmente solo se necesitan las variables KEYMAP y FONT. Por ejemplo, para una configuración polaca, se usaría:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="pl2"
FONT="lat2a-16 -m 8859-2"

# End /etc/sysconfig/console
EOF
```

• Como se mencionó anteriormente, a veces es necesario ajustar ligeramente el mapa de teclas estándar. El siguiente ejemplo añade el símbolo del euro al mapa de teclas alemán:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
FONT="lat0-16 -m 8859-15"
UNICODE="1"

# End /etc/sysconfig/console
EOF
```

• El siguiente es un ejemplo con Unicode habilitado para búlgaro, donde existe un mapa de teclas UTF-8 estándar:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="LatArCyrHeb-16"

# End /etc/sysconfig/console
EOF
```

• Debido al uso de un En el ejemplo anterior, la fuente LatArCyrHeb-16 de 512 glifos ya no permite colores brillantes en la consola de Linux a menos que se utilice un framebuffer. Si se desean colores brillantes sin framebuffer y se puede prescindir de caracteres que no pertenezcan al idioma, aún es posible usar una fuente específica del idioma de 256 glifos, como se ilustra a continuación:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="bg_bds-utf8"
FONT="cyr-sun16"

# End /etc/sysconfig/console
EOF
```

• El siguiente ejemplo ilustra la autoconversión del mapa de teclas de ISO-8859-15 a UTF-8 y la habilitación de teclas muertas en modo Unicode:

```
cat > /etc/sysconfig/console << "EOF"
# Begin /etc/sysconfig/console

UNICODE="1"
KEYMAP="de-latin1"
KEYMAP_CORRECTIONS="euro2"
LEGACY_CHARSET="iso-8859-15"
FONT="LatArCyrHeb-16 -m 8859-15"

# Fin de /etc/sysconfig/console
EOF
```

• Algunos mapas de teclas tienen teclas muertas (es decir, teclas que no producen un carácter por sí mismas, sino que acentúan el carácter producido por la siguiente tecla) o definen reglas de composición (como: "presione Ctrl+. A E para obtener Æ" en el mapa de teclas predeterminado). Linux-6.13.4 interpreta correctamente las teclas muertas y las reglas de composición del mapa de teclas solo cuando los caracteres de origen que se van a componer no son multibyte. Esta deficiencia no afecta a los mapas de teclado de los idiomas europeos, ya que en ellos se añaden acentos a los caracteres ASCII no acentuados o se combinan dos caracteres ASCII. Sin embargo, en el modo UTF-8 sí supone un problema; por ejemplo, en el griego, donde a veces es necesario acentuar la letra **α**. La solución es evitar el uso de UTF-8 o instalar el sistema X Window, que no presenta esta limitación, en su gestión de entrada.

• Para chino, japonés, coreano y otros idiomas, la consola de Linux no se puede configurar para mostrar los caracteres necesarios. Los usuarios que necesiten estos idiomas deben instalar el sistema X Window, fuentes que cubran los rangos de caracteres necesarios y el método de entrada adecuado (por ejemplo, SCIM admite una amplia variedad de idiomas).

Nota

El archivo */etc/sysconfig/console* solo controla la localización de la consola de texto de Linux. No tiene nada que ver con la configuración de la distribución de teclado ni las fuentes de terminal adecuadas en el sistema X Window, con sesiones ssh ni con una consola serie. En tales situaciones, las limitaciones mencionadas en los dos últimos puntos de la lista anterior no se aplican.

9.6.6. Creación de archivos durante el arranque

En ocasiones, es conveniente crear archivos durante el arranque. Por ejemplo, suele ser necesario el directorio */tmp/.ICE-unix*. Esto se puede hacer creando una entrada en el script de configuración */etc/sysconfig/createfiles*. El formato de este archivo está incrustado en los comentarios del archivo de configuración predeterminado.

9.6.7. Configuración del script Sysklogd

El script *sysklogd* invoca el programa **syslogd** como parte de la inicialización de System V. La opción -m 0 desactiva la marca de tiempo periódica que **syslogd** escribe en los archivos de registro cada 20 minutos de forma predeterminada. Para activar esta marca de tiempo periódica, edite */etc/sysconfig/rc.site* y defina la variable SYSKLOGD_PARMS con el valor deseado. Por ejemplo, para eliminar todos los parámetros, establezca la variable en un valor nulo:

```
SYSKLOGD_PARMS=
```

Consulte **man syslogd** para obtener más opciones.

9.6.8. El archivo rc.site

El archivo opcional */etc/sysconfig/rc.site* contiene la configuración que se establece automáticamente para cada script de arranque de System V. Alternativamente, puede establecer los valores especificados en los archivos de *nombre de host*, *consola* y *reloj* del directorio

/etc/sysconfig/. Si las variables asociadas están presentes tanto en estos archivos independientes como en *rc.site*, los valores de los archivos específicos del script surtirán efecto.

rc.site también contiene parámetros que permiten personalizar otros aspectos del proceso de arranque. Configurar la variable IPROMPT permitirá la ejecución selectiva de scripts de arranque. Otras opciones se describen en los comentarios del archivo. La versión predeterminada del archivo es la siguiente:

```
# rc.site
# Optional parameters for boot scripts.

# Distro Information
# These values, if specified here, override the defaults
#DISTRO="Linux From Scratch" # The distro name
#DISTRO_CONTACT="lfs-dev@lists.linuxfromscratch.org" # Bug report address
#DISTRO_MINI="LFS" # Short name used in filenames for distro config

# Define custom colors used in messages printed to the screen

# Please consult `man console_codes` for more information
# under the "ECMA-48 Set Graphics Rendition" section
#
# Warning: when switching from a 8bit to a 9bit font,
# the linux console will reinterpret the bold (1;) to
# the top 256 glyphs of the 9bit font. This does
# not affect framebuffer consoles

# These values, if specified here, override the defaults
#BRACKET="\\033[1;34m" # Blue
#FAILURE="\\033[1;31m" # Red
#INFO="\\033[1;36m"    # Cyan
#NORMAL="\\033[0;39m"  # Grey
#SUCCESS="\\033[1;32m" # Green
#WARNING="\\033[1;33m" # Yellow

# Use a colored prefix
# These values, if specified here, override the defaults
#BMPREFIX="      "
#SUCCESS_PREFIX="${SUCCESS}  *  ${NORMAL} "
#FAILURE_PREFIX="${FAILURE}*****${NORMAL} "
#WARNING_PREFIX="${WARNING} *** ${NORMAL} "

# Manually set the right edge of message output (characters)
# Useful when resetting console font during boot to override
# automatic screen width detection
#COLUMNS=120

# Interactive startup
#IPROMPT="yes" # Whether to display the interactive boot prompt
#itime="3"     # The amount of time (in seconds) to display the prompt

# The total length of the distro welcome string, without escape codes
#wlen=$(echo "Welcome to ${DISTRO}" | wc -c )
#welcome_message="Welcome to ${INFO}${DISTRO}${NORMAL}"

# The total length of the interactive string, without escape codes
#ilen=$(echo "Press 'I' to enter interactive startup" | wc -c )
```

```
#i_message="Press '${FAILURE}I${NORMAL}' to enter interactive startup"

# Set scripts to skip the file system check on reboot
#FASTBOOT=yes

# Skip reading from the console
#HEADLESS=yes

# Write out fsck progress if yes
#VERBOSE_FSCK=no

# Speed up boot without waiting for settle in udev
#OMIT_UDEV_SETTLE=y

# Speed up boot without waiting for settle in udev_retry
#OMIT_UDEV_RETRY_SETTLE=yes

# Skip cleaning /tmp if yes
#SKIPTMPCLEAN=no

# For setclock
#UTC=1
#CLOCKPARAMS=

# For consolelog (Note that the default, 7=debug, is noisy)
#LOGLEVEL=7

# For network
#HOSTNAME=mylfs

# Delay between TERM and KILL signals at shutdown
#KILLDELAY=3

# Optional sysklogd parameters
#SYSKLOGD_PARMS="-m 0"

# Console parameters
#UNICODE=1
#KEYMAP="de-latin1"
#KEYMAP_CORRECTIONS="euro2"
#FONT="lat0-16 -m  8859-15"
#LEGACY_CHARSET=
```

9.6.8.1. Personalización de los scripts de arranque y apagado

Los scripts de arranque LFS arrancan y apagan un sistema de forma bastante eficiente, pero puede realizar algunos ajustes en el archivo rc.site para mejorar aún más la velocidad y ajustar los mensajes según sus preferencias. Para ello, ajuste la configuración en el archivo *`/etc/sysconfig/rc.site`* mencionado anteriormente.

• Durante el script de arranque *`udev`*, se realiza una llamada a **udev settle** que requiere un tiempo para completarse. Este tiempo puede ser necesario o no según los dispositivos del sistema. Si solo tiene particiones simples y una sola tarjeta Ethernet, el proceso de arranque probablemente no necesite esperar este comando. Para omitirlo, configure la variable OMIT_UDEV_SETTLE=y.

• El script de arranque `udev_retry` también ejecuta **udev settle** por defecto. Este comando solo es necesario si el directorio */var* se monta por separado, ya que el reloj necesita el archivo */var/lib/hwclock/adjtime*. Otras personalizaciones también pueden requerir esperar a que udev se complete, pero en muchas instalaciones no es necesario. Omita el comando configurando la variable OMIT_UDEV_RETRY_SETTLE=y.

• De forma predeterminada, las comprobaciones del sistema de archivos son silenciosas. Esto puede parecer un retraso durante el proceso de arranque. Para activar la salida **fsck**, configure la variable VERBOSE_FSCK=y.

• Al reiniciar, puede que desee omitir por completo la comprobación del sistema de archivos, **fsck**. Para ello, cree el archivo */fastboot* o reinicie el sistema con el comando **/sbin/shutdown -f -r now**. Por otro lado, puede forzar la comprobación de todos los sistemas de archivos creando */forcefsck* o ejecutando el comando **shutdown** con el parámetro *-F* en lugar de *-f*.

Al configurar la variable FASTBOOT=y, se deshabilitará **fsck** durante el proceso de arranque hasta que se elimine. No se recomienda hacerlo de forma permanente.

• Normalmente, todos los archivos del directorio `/tmp` se eliminan al arrancar. Dependiendo de la cantidad de archivos o directorios presentes, esto puede causar un retraso considerable en el proceso de arranque. Para omitir la eliminación de estos archivos, configure la variable SKIPTMPCLEAN=y.

• Durante el apagado, el programa init envía una señal TERM a cada programa que ha iniciado (p. ej., agetty), espera un tiempo determinado (3 segundos por defecto), envía a cada proceso una señal KILL y vuelve a esperar. Este proceso se repite en el script **sendsignals** para cualquier proceso que no se apague con sus propios scripts. El retraso de **init** se puede configurar pasando un parámetro. Por ejemplo, para eliminar el retraso de **init**, pase el parámetro -t0 al apagar o reiniciar (p. ej., **`/sbin/shutdown -t0 -r now`**). El retraso del script **sendsignals** se puede omitir configurando el parámetro KILLDELAY=0.

9.7. Configuración de la configuración regional del sistema

Algunas variables de entorno son necesarias para la compatibilidad con el idioma nativo. Una configuración correcta da como resultado:

• La traducción de la salida de los programas a su idioma nativo.

• La clasificación correcta de caracteres en letras, dígitos y otras clases. Esto es necesario para que bash acepte correctamente caracteres no ASCII en líneas de comandos en configuraciones regionales distintas del inglés.

• El orden alfabético correcto para el país.

• El tamaño de papel predeterminado adecuado.

• El formato correcto de los valores monetarios, de hora y de fecha.

Reemplace `<ll>` a continuación con el código de dos letras del idioma deseado (p. ej., en) y `<CC>` con el código de dos letras del país correspondiente (p. ej., GB). `<charmap>` debe reemplazarse con el mapa de caracteres canónico de la configuración regional elegida.

También pueden estar presentes modificadores opcionales como `@euro`. La lista de todas las configuraciones regionales compatibles con Glibc se puede obtener ejecutando el siguiente comando:

```
locale -a
```

Charmaps puede tener varios alias; por ejemplo, *ISO-8859-1* también se conoce como *iso8859-1* e *iso88591*. Algunas aplicaciones no pueden gestionar correctamente los sinónimos (p. ej., requieren que *UTF-8* se escriba como *UTF-8*, no como *utf8*), por lo que, en la mayoría de los casos, lo más seguro es elegir el nombre canónico para una configuración regional en particular. Para determinar el nombre canónico, ejecute el siguiente comando, donde *<nombre de configuración regional>* es el resultado de **locale -a** para su configuración regional preferida (*en_GB*. *iso88591* en nuestro ejemplo).

```
LC_ALL=<locale name> locale charmap
```

Para la configuración regional *en_GB.iso88591*, el comando anterior imprimirá:

```
ISO-8859-1
```

Esto da como resultado una configuración regional final de *en_GB.ISO-8859-1*. Es importante que la configuración regional encontrada mediante la heurística anterior se pruebe antes de agregarla a los archivos de inicio de Bash:

```
LC_ALL=<locale name> locale language
LC_ALL=<locale name> locale charmap
LC_ALL=<locale name> locale int_curr_symbol
LC_ALL=<locale name> locale int_prefix
```

Los comandos anteriores deben imprimir el nombre del idioma, la codificación de caracteres utilizada por la configuración regional, la moneda local y el prefijo que se debe marcar antes del número de

teléfono para ingresar al país. Si alguno de los comandos anteriores falla con un mensaje similar al que se muestra a continuación, significa que su configuración regional no se instaló en el Capítulo 8 o no es compatible con la instalación predeterminada de Glibc.

```
locale: Cannot set LC_* to default locale: No such file or directory
```

Si esto ocurre, debe instalar la configuración regional deseada mediante el comando **localedef** o considerar elegir una configuración regional diferente. Las instrucciones posteriores asumen que no hay tales mensajes de error de Glibc.

Otros paquetes también pueden funcionar incorrectamente (aunque no necesariamente mostrarán ningún mensaje de error) si el nombre de la configuración regional no cumple con sus expectativas. En esos casos, investigar cómo otras distribuciones de Linux admiten su configuración regional podría proporcionar información útil.

El programa de shell **/bin/bash** (en adelante, "el shell") utiliza una colección de archivos de inicio para ayudar a crear el entorno de ejecución. Cada archivo tiene un uso específico y puede afectar los entornos de inicio de sesión e interactivos de forma diferente. Los archivos del directorio *`/etc`* proporcionan la configuración global. Si existen archivos equivalentes en el directorio de inicio, pueden anular la configuración global.

Un shell de inicio de sesión interactivo se inicia tras un inicio de sesión exitoso mediante **/bin/login**, leyendo el archivo */etc/passwd*. Un shell interactivo sin inicio de sesión se inicia en la línea de comandos (p. ej., *`[prompt]$`* **/bin/bash**). Un shell no interactivo suele estar presente cuando se ejecuta un script de shell. No es interactivo porque procesa un script y no espera la entrada del usuario entre comandos.

Cree el archivo *`/etc/profile`* una vez que se hayan determinado las configuraciones regionales correctas para establecer la configuración regional deseada, pero configure la configuración regional *`C.UTF-8`* si se ejecuta en la consola de Linux (para evitar que los programas muestren caracteres que la consola de Linux no puede representar).

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

for i in $(locale); do
  unset ${i%=*}
done

if [[ "$TERM" = linux ]]; then
  export LANG=C.UTF-8
else
  export LANG=<ll>_<CC>.<charmap><@modifiers>
fi

# End /etc/profile
EOF
```

Las configuraciones regionales *`C`* (predeterminada) y *`en_US`* (recomendada para usuarios de inglés estadounidense) son diferentes. *`C`* utiliza el conjunto de caracteres US-ASCII de 7 bits y trata los bytes

con el bit alto establecido como caracteres no válidos. Por eso, por ejemplo, el comando **ls** los sustituye por signos de interrogación en esa configuración regional. Además, intentar enviar correo con dichos caracteres desde Mutt o Pine da como resultado el envío de mensajes no conformes con RFC (el conjunto de caracteres del correo saliente se indica como *unknown 8-bit*). Se recomienda usar la configuración regional *C* solo si está seguro de que nunca necesitará caracteres de 8 bits.

9.8. Creación del archivo /etc/inputrc

El archivo *inputrc* es el archivo de configuración de la biblioteca readline, que proporciona funciones de edición mientras el usuario introduce una línea desde la terminal. Funciona traduciendo las entradas del teclado en acciones específicas. Readline es utilizado por bash y la mayoría de los demás shells, así como por muchas otras aplicaciones.

La mayoría de los usuarios no necesitan funciones específicas, por lo que el siguiente comando crea un archivo */etc/inputrc* global que usarán todos los usuarios que inicien sesión. Si posteriormente decide que necesita anular los valores predeterminados para cada usuario, puede crear un archivo *.inputrc* en el directorio personal del usuario con las asignaciones modificadas.

Para obtener más información sobre cómo editar el archivo *inputrc*, consulte "**info bash**" en la sección "*Archivo de inicio de Readline*". **info readline** también es una buena fuente de información.

A continuación se muestra un archivo *inputrc* global genérico junto con comentarios que explican la función de las distintas opciones. Tenga en cuenta que los comentarios no pueden estar en la misma línea que los comandos. Cree el archivo con el siguiente comando:

```
cat > /etc/inputrc << "EOF"
# Begin /etc/inputrc
# Modified by Chris Lynn <roryo@roryo.dynup.net>

# Allow the command prompt to wrap to the next line
set horizontal-scroll-mode Off

# Enable 8-bit input
set meta-flag On
set input-meta On

# Turns off 8th bit stripping
set convert-meta Off

# Keep the 8th bit for display
set output-meta On

# none, visible or audible
set bell-style none

# All of the following map the escape sequence of the value
# contained in the 1st argument to the readline specific functions
"\eOd": backward-word
"\eOc": forward-word

# for linux console
"\e[1~": beginning-of-line
"\e[4~": end-of-line
"\e[5~": beginning-of-history
"\e[6~": end-of-history
"\e[3~": delete-char
"\e[2~": quoted-insert

# for xterm
"\eOH": beginning-of-line
"\eOF": end-of-line

# for Konsole
"\e[H": beginning-of-line
"\e[F": end-of-line

# End /etc/inputrc
EOF
```

9.9. Creación del archivo /etc/shells

El archivo *shells* contiene una lista de shells de inicio de sesión en el sistema. Las aplicaciones utilizan este archivo para determinar si un shell es válido. Para cada shell, debe haber una sola línea con la ruta del shell relativa a la raíz de la estructura de directorios (/).

Por ejemplo, **chsh** consulta este archivo para determinar si un usuario sin privilegios puede cambiar el shell de inicio de sesión de su propia cuenta. Si el nombre del comando no aparece en la lista, se le denegará al usuario la posibilidad de cambiar de shell.

Es un requisito para aplicaciones como GDM, que no rellena el explorador de accesos si no encuentra */etc/shells*, o para los daemons FTP, que tradicionalmente impiden el acceso a usuarios con shells no incluidas en este archivo.

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

/bin/sh
/bin/bash

# End /etc/shells
EOF
```

Capítulo 10. Haciendo que el sistema LFS sea arrancable

10.1. Introducción

Ha llegado el momento de hacer que el sistema LFS sea arrancable. Este capítulo explica la creación del archivo /etc/fstab, la compilación de un kernel para el nuevo sistema LFS y la instalación del gestor de arranque GRUB para que el sistema LFS pueda seleccionarse para el arranque al inicio.

10.2. Creación del archivo /etc/fstab

Algunos programas utilizan el archivo /etc/fstab para determinar dónde se montarán los sistemas de archivos por defecto, en qué orden y cuáles deben comprobarse (para detectar errores de integridad) antes del montaje. Cree una nueva tabla de sistemas de archivos como la siguiente:

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

# file system  mount-point      type       options              dump  fsck order
/dev/<xxx>      /               <fff>      defaults               1       1
/dev/<yyy>       swap           swap       pri=1                  0       0
proc            /proc           proc       nosuid,noexec,nodev    0       0
sysfs           /sys            sysfs      nosuid,noexec,nodev    0       0
devpts          /dev/pts        devpts     gid=5,mode=620         0       0
tmpfs           /run            tmpfs      defaults               0       0
devtmpfs        /dev            devtmpfs   mode=0755,nosuid       0       0
tmpfs           /dev/shm        tmpfs      nosuid,nodev           0       0
cgroup2         /sys/fs/cgroup  cgroup2    nosuid,noexec,nodev    0       0

# End /etc/fstab
EOF
```

Reemplace <*xxx*>, <*yyy*> y <*fff*> con los valores apropiados para el sistema, por ejemplo, *sda2*, *sda5* y ext4. Para obtener más información sobre los seis campos de este archivo, consulte *fstab(5)*.

Los sistemas de archivos con origen en MS-DOS o Windows (p. ej., vfat, ntfs, smbfs, cifs, iso9660, udf) necesitan una opción especial, utf8, para que los caracteres no ASCII en los nombres de archivo se interpreten correctamente. Para configuraciones regionales que no sean UTF-8, el valor de *iocharset* debe ser el mismo que el conjunto de caracteres de la configuración regional, ajustado de forma que el kernel lo comprenda. Esto funciona si la definición del conjunto de caracteres correspondiente (que se encuentra en Sistemas de archivos -> Compatibilidad con idiomas nativos al configurar el kernel) se ha compilado en el kernel o se ha creado como módulo. Sin embargo, si el conjunto de caracteres de la configuración regional es UTF-8, la opción correspondiente *iocharset=utf8* haría que el sistema de archivos distinga entre mayúsculas y minúsculas. Para solucionar esto, utilice la opción especial utf8 en lugar de *iocharset=utf8* para las configuraciones regionales UTF-8. La opción "codepage" también es necesaria para los sistemas de archivos vfat y smbfs. Debe configurarse con el número de página de códigos utilizado en MS-DOS en su país. Por ejemplo, para montar unidades flash USB, un usuario ru_RU.KOI8-R necesitaría lo siguiente en la sección de opciones de su línea de montaje en */etc/fstab*:

```
noauto,user,quiet,showexec,codepage=866,iocharset=koi8r
```

El fragmento de opciones correspondiente para los usuarios ru_RU.UTF-8 es:

```
noauto,user,quiet,showexec,codepage=866,utf8
```

Tenga en cuenta que usar *iocharset* es la opción predeterminada para *iso8859-1* (que evita que el sistema de archivos distinga entre mayúsculas y minúsculas), y la opción *utf8* indica al kernel *que convierta* los nombres de archivo a UTF-8 para que puedan interpretarse en la configuración regional UTF-8.

También es posible especificar la página de códigos predeterminada y los valores de iocharset para algunos sistemas de archivos durante la configuración del kernel. Los parámetros relevantes se denominan "Opción NLS predeterminada" (CONFIG_NLS_DEFAULT), "Opción NLS remota predeterminada" (CONFIG_SMB_NLS_DEFAULT), "Página de códigos predeterminada para FAT" (CONFIG_FAT_DEFAULT_CODEPAGE) y "Conjunto de iocharset predeterminado para FAT" (CONFIG_FAT_DEFAULT_IOCHARSET). No es posible especificar estas configuraciones para el sistema de archivos NTFS durante la compilación del kernel.

Es posible que el sistema de archivos ext3 sea fiable durante cortes de energía para algunos tipos de discos duros. Para ello, agregue la opción de montaje *barrier=1* a la entrada correspondiente en */etc/fstab*. Para comprobar si la unidad de disco admite esta opción, ejecute *hdparm* en la unidad de disco correspondiente. Por ejemplo, si:

```
hdparm -I /dev/sda | grep NCQ
```

devuelve una salida no vacía; la opción es compatible.

Nota: Las particiones basadas en la gestión de volúmenes lógicos (LVM) no pueden usar la opción de *barrier*.

10.3. Linux-6.13.4

El paquete Linux contiene el kernel de Linux.

Tiempo de compilación aproximado: 0,4 - 32 SBU (normalmente unas 2,5 SBU)
Espacio en disco requerido: 1,7 - 14 GB (normalmente unos 2,3 GB)

10.3.1. Instalación del kernel

La compilación del kernel implica varios pasos: configuración, compilación e instalación. Consulte el archivo README en el árbol de código fuente del kernel para conocer métodos alternativos a la configuración del kernel en este libro.

Importante

Compilar el kernel de Linux por primera vez es una de las tareas más difíciles en LFS. Su correcta configuración depende del hardware específico del sistema de destino y de sus necesidades específicas. Hay casi 12 000 elementos de configuración disponibles para el kernel, aunque solo un tercio de ellos son necesarios para la mayoría de los ordenadores. Los editores de LFS recomiendan a los usuarios que no estén familiarizados con este proceso que sigan atentamente los procedimientos que se indican a continuación. El objetivo es que el sistema inicial pueda iniciar sesión desde la línea de comandos al reiniciar, como se explica más adelante en la Sección 11.3, "Reinicio del sistema". En este punto, la optimización y la personalización no son un objetivo.

Para obtener información general sobre la configuración del kernel, consulte https://www.linuxfromscratch.org/hints/downloads/files/kernel-configuration.txt. Puede encontrar información adicional sobre la configuración y la compilación del kernel en https://anduin.linuxfromscratch.org/LFS/kernel-nutshell/.
Estas referencias están un poco desactualizadas, pero ofrecen una visión general del proceso.

Si todo lo demás falla, puede solicitar ayuda en la lista de correo lfs-support. Tenga en cuenta que es necesario suscribirse para evitar el spam.

Prepárese para la compilación ejecutando el siguiente comando:

```
make mrproper
```

Esto garantiza que el árbol del kernel esté completamente limpio. El equipo del kernel recomienda ejecutar este comando antes de cada compilación del kernel. No confíe en que el árbol de fuentes esté limpio después de descomprimir.

Hay varias maneras de configurar las opciones del kernel. Normalmente, esto se hace mediante una interfaz de menús, por ejemplo:

```
make menuconfig
```

Significado de las variables de entorno opcionales de make:

LANG=<host_LANG_value> LC_ALL=
: Esto establece la configuración regional a la utilizada en el host. Esto puede ser necesario para un correcto dibujo de líneas de la interfaz de ncurses de menuconfig en una consola de texto UTF-8 de Linux.

Si se utiliza, asegúrese de reemplazar *<host_LANG_value>* por el valor de la variable *$LANG* de su host. Alternativamente, puede usar el valor del host *$LC_ALL* o *$LC_CTYPE*.

make menuconfig

Esto inicia una interfaz de ncurses basada en menús. Para otras interfaces (gráficas), escriba **make help**.

Nota

Un buen punto de partida para configurar el kernel es ejecutar **make defconfig**. Esto establecerá la configuración base en un estado óptimo que tenga en cuenta la arquitectura actual de su sistema.
Asegúrese de habilitar, deshabilitar o configurar las siguientes funciones; de lo contrario, el sistema podría no funcionar correctamente o no arrancar:

```
General setup --->
  [ ] Compile the kernel with warnings as errors                                [WERROR]
  CPU/Task time and stats accounting --->
    [*] Pressure stall information tracking                                       [PSI]
    [ ]   Require boot parameter to enable pressure stall information tracking
                                                          ... [PSI_DEFAULT_DISABLED]
  < > Enable kernel headers through /sys/kernel/kheaders.tar.xz               [IKHEADERS]
  [*] Control Group support --->                                                [CGROUPS]
  [*] Memory controller                                                           [MEMCG]
  [ ] Configure standard kernel features (expert users) --→                     [EXPERT]

Processor type and features --→
  [*] Build a relocatable kernel                                          [RELOCATABLE]
  [*]    Randomize the address of the kernel image (KASLR)               [RANDOMIZE_BASE]

General architecture-dependent options --->
[*] Stack Protector buffer overflow detection                           [STACKPROTECTOR]
[*]   Strong Stack Protector                                  [STACKPROTECTOR_STRONG]

Device Drivers --->
  Generic Driver Options --->
    [ ] Support for uevent helper                                        [UEVENT_HELPER]
    [*] Maintain a devtmpfs filesystem to mount at /dev                       [DEVTMPFS]
    [*]   Automount devtmpfs at /dev, after the kernel mounted the rootfs
                                                                ... [DEVTMPFS_MOUNT]
  Firmware Drivers --->
    [*] Mark VGA/VBE/EFI FB as generic system framebuffer                  [SYSFB_SIMPLEFB]
  Graphics support --->
    <*> Direct Rendering Manager (XFree86 4.1.0 and higher DRI support) --->
                                                                              ... [DRM]
      [*]    Display a user-friendly message when a kernel panic occurs
                                                                         ... [DRM_PANIC]
      (kmsg)    Panic screen formatter                                [DRM_PANIC_SCREEN]
      Supported DRM clients --->
        [*] Enable legacy fbdev support for your modesetting driver
                                                              ... [DRM_FBDEV_EMULATION]
      <*>    Simple framebuffer driver                                    [DRM_SIMPLEDRM]
Console display driver support --->
      [*] Framebuffer Console support                          [FRAMEBUFFER_CONSOLE]
```

Habilite algunas funciones adicionales si está construyendo un sistema de 64 bits. Si usa menuconfig, habilítelas en el orden *CONFIG_PCI_MSI* primero, luego CONFIG_IRQ_REMAP y, por último, *CONFIG_X86_X2APIC*, ya que una opción solo aparece después de seleccionar sus dependencias.

```
Processor type and features --->
  [*] Support x2apic                                                      [X86_X2APIC]
```

```
Device Drivers --->
  [*] PCI support --→                                                  [PCI]
    [*] Message Signaled Interrupts (MSI and MSI-X)                [PCI_MSI]
  [*] IOMMU Hardware Support --->                                [IOMMU_SUPPORT]
    [*] Support for Interrupt Remapping                             [IRQ_REMAP]
```

Si está creando un sistema de 32 bits con hardware con más de 4 GB de RAM, ajuste la configuración para que el kernel pueda usar hasta 64 GB de RAM física:

```
Processor type and features --->
  High Memory Support --->
    (X) 64GB                                                      [HIGHMEM64G]
```

Si la partición del sistema LFS está en un SSD NVME (es decir, el nodo de dispositivo de la partición es /dev/nvme* en lugar de /dev/sd*), habilite la compatibilidad con NVME; de lo contrario, el sistema LFS no arrancará:

```
Device Drivers --->
  NVME Support --->
    <*> NVM Express block device                                 [BLK_DEV_NVME]
```

Existen otras opciones que podrían ser útiles según los requisitos del sistema. Para obtener una lista de las opciones necesarias para los paquetes BLFS, consulte el *Índice de configuración del kernel de BLFS*.

Nota

Si el hardware de su host usa UEFI y desea arrancar el sistema LFS con él, debe ajustar la configuración del kernel siguiendo la página de *BLFS*, **incluso si va a usar el gestor de arranque UEFI de la distribución del host**.

Razones para los elementos de configuración anteriores:

Aleatorizar la dirección de la imagen del kernel (KASLR)
Randomize the address of the kernel image (KASLR)

Habilitar ASLR para la imagen del kernel para mitigar algunos ataques basados en direcciones fijas de datos o código confidencial en el kernel.

Compilar el kernel con advertencias como errores
Compile the kernel with warnings as errors

Esto puede causar un fallo de compilación si el compilador o la configuración son diferentes a los de los desarrolladores del kernel.

Habilitar los encabezados del kernel a través de /sys/kernel/kheaders.tar.xz
Enable kernel headers through /sys/kernel/kheaders.tar.xz

Esto requerirá **cpio** para compilar el kernel. LFS no instala **cpio**.

Configurar las funciones estándar del kernel (usuarios expertos)
Configure standard kernel features (expert users)

Esto hará que aparezcan algunas opciones en la interfaz de configuración, pero modificarlas puede ser peligroso. No utilice esta opción a menos que sepa lo que hace.

Potente protector de pila
Strong Stack Protector

Habilite SSP para el kernel. Lo hemos habilitado para todo el espacio de usuario con –enable-default-ssp al configurar GCC, pero el kernel no utiliza la configuración predeterminada de GCC para SSP. Lo habilitamos explícitamente aquí.

```
Compatibilidad con el asistente uevent
Support for uevent helper
```

Tener esta opción activada puede interferir con la administración de dispositivos al usar Udev.

```
Mantener un devtmpfs
Maintain a devtmpfs
```

Esto creará nodos de dispositivo automatizados que se rellenan con el kernel, incluso sin que Udev se esté ejecutando. Udev se ejecuta sobre él, administrando permisos y añadiendo enlaces simbólicos. Esta configuración es necesaria para todos los usuarios de Udev.

```
Montar automáticamente devtmpfs en /dev
Automount devtmpfs at /dev
```

Esto montará la vista del kernel de los dispositivos en /dev al cambiar al sistema de archivos raíz justo antes de iniciar init.

```
Mostrar un mensaje intuitivo cuando se produce un pánico del kernel
Display a user-friendly message when a kernel panic occurs
```

Esto hará que el kernel muestre correctamente el mensaje en caso de un pánico del kernel, siempre que un controlador DRM en ejecución lo permita. Sin esto, sería más difícil diagnosticar un pánico: si no se está ejecutando ningún controlador DRM, estaríamos en una consola VGA que solo tiene capacidad para 24 líneas y el mensaje del kernel relevante suele desaparecer; si se está ejecutando un controlador DRM, la pantalla suele quedar completamente bloqueada en caso de pánico. A partir de Linux-6.12, ninguno de los controladores dedicados para los modelos de GPU convencionales es compatible con esto, pero sí lo es el "controlador de framebuffer simple", que se ejecuta en el framebuffer VESA (o EFI) antes de cargar el controlador de GPU dedicado. Si el controlador de GPU dedicado se crea como un módulo (en lugar de una parte de la imagen del kernel) y no se utiliza initramfs, esta funcionalidad funcionará bien antes de que se monte el sistema de archivos raíz y ya es suficiente para brindar información sobre la mayoría de los errores de configuración de LFS que causan un pánico (por ejemplo, una configuración root= incorrecta en la Sección 10.4, "Uso de GRUB para configurar el proceso de arranque").

```
Formateador de pantalla de pánico
Panic screen formatter
```

Configure este kmsg para asegurar que se muestren las últimas líneas de mensajes del kernel cuando se produce un pánico del kernel. La opción predeterminada, `user`, haría que el kernel solo mostrara un mensaje de pánico intuitivo, lo cual no es útil para el diagnóstico. La tercera opción, `qr_code`, haría que el kernel comprimiera las últimas líneas de mensajes del kernel en un código QR y lo mostrara. El código QR puede contener más líneas de mensaje que texto plano y se puede decodificar con un dispositivo externo (como un smartphone). Sin embargo, requiere un compilador de Rust que LFS no proporciona.

Marcar el FB VGA/VBE/EFI como framebuffer genérico del sistema y controlador de framebuffer simple
Mark VGA/VBE/EFI FB as generic system framebuffer and Simple framebuffer driver

Estos permiten usar el framebuffer VESA (o el framebuffer EFI si se inicia el sistema LFS mediante UEFI) como dispositivo DRM. El framebuffer VESA se configurará mediante GRUB (o el framebuffer EFI se configurará mediante el firmware UEFI), por lo que el controlador de pánico DRM puede funcionar antes de que se cargue el controlador DRM específico de la GPU.

Habilite la compatibilidad con fbdev heredado para su controlador de configuración de modo y la compatibilidad con la consola Framebuffer
Enable legacy fbdev support for your modesetting driver and Framebuffer Console support

Estos son necesarios para mostrar la consola Linux en una GPU controlada por un controlador DRI (Infraestructura de Renderizado Directo). Como CONFIG_DRM (Administrador de Renderizado Directo) está habilitado, también debemos habilitar estas dos opciones; de lo contrario, veremos una pantalla en blanco una vez cargado el controlador DRI.

Compatibilidad con x2apic
Support x2apic

Compatibilidad con la ejecución del controlador de interrupciones de procesadores x86 de 64 bits en modo x2APIC. x2APIC puede estar habilitado por firmware en sistemas x86 de 64 bits, y un kernel sin esta opción habilitada generará un error al arrancar si x2APIC está habilitado por firmware. Esta opción no tiene ningún efecto, pero tampoco es perjudicial si x2APIC está deshabilitado por firmware.

Como alternativa, **make oldconfig** puede ser más adecuado en algunas situaciones. Consulte el archivo README para obtener más información.

Si lo desea, omita la configuración del kernel copiando el archivo de configuración del kernel, `.config`, desde el sistema host (siempre que esté disponible) al directorio *`linux-6.13.4`* descomprimido. Sin embargo, no recomendamos esta opción. Suele ser mejor explorar todos los menús de configuración y crear la configuración del kernel desde cero.

Compilar la imagen del kernel y los módulos:

```
make
```

Si utiliza módulos del kernel, puede que se requiera la configuración del módulo en */etc/modprobe.d*. La información sobre los módulos y la configuración del kernel se encuentra en la Sección 9.3, “Descripción general del manejo de dispositivos y módulos” y en la documentación del kernel, en el directorio *`linux-6.13.4/Documentation`*. También puede ser de interés el archivo *modprobe.d(5)*.

A menos que se haya deshabilitado la compatibilidad con módulos en la configuración del kernel, instale los módulos con:

```
make modules_install
```

Una vez completada la compilación del kernel, se requieren pasos adicionales para completar la instalación. Algunos archivos deben copiarse al directorio `/boot`.

Precaución

Si ha decidido usar una partición /boot independiente para el sistema LFS (quizás compartiendo una partición /boot con la distribución del host), los archivos copiados a continuación deberían ir allí. La forma más sencilla de hacerlo es crear primero la entrada para /boot en /etc/fstab (consulte la sección anterior para obtener más información) y, a continuación, ejecutar el siguiente comando como usuario root en el *entorno chroot*:

```
mount /boot
```

La ruta al nodo del dispositivo se omite en el comando porque **mount** puede leerla desde */etc/fstab*.

La ruta a la imagen del kernel puede variar según la plataforma utilizada. El nombre de archivo a continuación se puede modificar según sus preferencias, pero la raíz del nombre de archivo debe ser "vmlinuz" para que sea compatible con la configuración automática del proceso de arranque descrito en la siguiente sección. El siguiente comando asume una arquitectura x86:

```
cp -iv arch/x86/boot/bzImage /boot/vmlinuz-6.13.4-lfs-12.3
```

System.map es un archivo de símbolos para el kernel. Mapea los puntos de entrada de cada función en la API del kernel, así como las direcciones de las estructuras de datos del kernel en ejecución. Se utiliza como recurso para investigar problemas del kernel. Ejecute el siguiente comando para instalar el archivo de mapa:

```
cp -iv System.map /boot/System.map-6.13.4
```

El archivo de configuración del kernel *.config* generado por el paso "**make menuconfig**" anterior contiene todas las selecciones de configuración para el kernel que se acaba de compilar. Es recomendable guardar este archivo para futuras referencias:

```
cp -iv .config /boot/config-6.13.4
```

Instalar la documentación del kernel de Linux:

```
cp -r Documentation -T /usr/share/doc/linux-6.13.4
```

Es importante tener en cuenta que los archivos del directorio de origen del kernel no son propiedad del usuario root. Al descomprimir un paquete como usuario root (como hicimos en chroot), los archivos tienen los ID de usuario y grupo del equipo del empaquetador. Esto no suele ser un problema para la instalación de otros paquetes, ya que el árbol de origen se elimina después de la instalación. Sin embargo, el árbol de origen de Linux suele conservarse durante mucho tiempo. Por ello, existe la posibilidad de que el ID de usuario utilizado por el empaquetador se asigne a alguien en el equipo. Esa persona tendría entonces acceso de escritura al código fuente del kernel.

Nota

En muchos casos, será necesario actualizar la configuración del kernel para los paquetes que se instalarán posteriormente en BLFS. A diferencia de otros paquetes, no es necesario eliminar el árbol de fuentes del kernel después de instalar el nuevo kernel.

Si se va a conservar el árbol de fuentes del kernel, ejecute **chown -R 0:0** en el directorio *linux-6.13.4* para garantizar que todos los archivos pertenezcan al usuario *root*.

Si está actualizando la configuración y reconstruyendo el kernel a partir de un árbol de fuentes del kernel conservado, normalmente **no** debería ejecutar el comando **make mrproper**. Este comando purgaría el archivo *.config* y todos los archivos .o de la compilación anterior. Aunque es fácil restaurar el archivo *.config* desde la copia en */boot*, purgar todos los archivos .o sigue siendo un desperdicio: para un cambio de configuración simple, a menudo solo es necesario (re)compilar unos pocos archivos .o y el sistema de compilación del kernel omitirá correctamente los demás archivos .o si no se purgan. Por otro lado, si ha actualizado GCC, debería ejecutar "**make clean**" para purgar todos los archivos .o de la compilación anterior; de lo contrario, la nueva compilación podría fallar.

Advertencia

Alguna documentación del kernel recomienda crear un enlace simbólico desde */usr/src/linux* que apunte al directorio fuente del kernel. Esto es específico para kernels anteriores a la serie 2.6 y *no debe* crearse en un sistema LFS, ya que puede causar problemas con los paquetes que desee compilar una vez que su sistema LFS base esté completo.

10.3.2. Configuración del orden de carga de módulos de Linux

La mayoría de las veces, los módulos de Linux se cargan automáticamente, pero a veces se requiere una instrucción específica. El programa que carga los módulos, **modprobe** o **insmod**, utiliza */etc/modprobe.d/usb.conf* para este propósito. Este archivo debe crearse para que, si los controladores USB (ehci_hcd, ohci_hcd y uhci_hcd) se han compilado como módulos, se carguen en el orden correcto. ehci_hcd debe cargarse antes que ohci_hcd y uhci_hcd para evitar una advertencia al arrancar.

Cree un nuevo archivo */etc/modprobe.d/usb.conf* ejecutando lo siguiente:

```
install -v -m755 -d /etc/modprobe.d
cat > /etc/modprobe.d/usb.conf << "EOF"
# Begin /etc/modprobe.d/usb.conf

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

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

10.3.3. Contenido de Linux

Archivos instalados: config-6.13.4, vmlinuz-6.13.4-lfs-12.3 y System.map-6.13.4

Directorios instalados: /lib/modules, /usr/share/doc/linux-6.13.4

Descripciones breves

`config-6.13.4` Contiene todas las opciones de configuración del kernel.

`vmlinuz-6.13.4-lfs-12.3`	El motor del sistema Linux. Al encender el ordenador, el kernel es la primera parte del sistema operativo que se carga. Detecta e inicializa todos los componentes del hardware del ordenador, los pone a disposición del software como un árbol de archivos y convierte una sola CPU en una máquina multitarea capaz de ejecutar numerosos programas prácticamente al mismo tiempo.
`System.map-6.13.4`	Una lista de direcciones y símbolos. Mapea los puntos de entrada y las direcciones de todas las funciones y estructuras de datos en el núcleo.

10.4. Uso de GRUB para configurar el proceso de arranque

Nota

Si su sistema es compatible con UEFI y desea arrancar LFS con UEFI, debe omitir las instrucciones de esta página, pero aun así debe aprender la sintaxis de *grub.cfg* y el método para especificar una partición en el archivo desde esta página, y configurar GRUB con compatibilidad con UEFI siguiendo las instrucciones de la *página BLFS*.

10.4.1. Introducción

Advertencia

Configurar GRUB incorrectamente puede dejar su sistema inoperativo sin un dispositivo de arranque alternativo, como un CD-ROM o una unidad USB de arranque. Esta sección no es necesaria para arrancar su sistema LFS. Quizás simplemente desee modificar su gestor de arranque actual, por ejemplo, Grub-Legacy, GRUB2 o LILO.

Asegúrese de tener listo un disco de arranque de emergencia para rescatar el equipo si este queda inutilizable (no arrancable). Si aún no tiene un dispositivo de arranque, puede crear uno. Para que el siguiente procedimiento funcione, debe acceder a BLFS e instalar **xorriso** desde el paquete *libisoburn*.

```
cd /tmp
grub-mkrescue --output=grub-img.iso
xorriso -as cdrecord -v dev=/dev/cdrw blank=as_needed grub-img.iso
```

10.4.2. Convenciones de nomenclatura de GRUB

GRUB utiliza su propia estructura de nomenclatura para unidades y particiones: *(hdn,m)*, donde *n* es el número del disco duro y *m* es el número de la partición. Los números de los discos duros empiezan desde cero, pero los números de las particiones empiezan desde uno para las particiones normales (desde cinco para las particiones extendidas). Tenga en cuenta que esto difiere de las versiones anteriores, donde ambos números empezaban desde cero. Por ejemplo, la partición *sda1* es *(hd0,1)* para GRUB y *sdb3* es *(hd1,3)*. A diferencia de Linux, GRUB no considera las unidades de CD-ROM como discos duros. Por ejemplo, si se usa un CD en *hdb* y un segundo disco duro en *hdc*, ese segundo disco duro seguirá siendo *(hd1)*.

10.4.3. Configuración

GRUB funciona escribiendo datos en la primera pista física del disco duro. Esta área no forma parte de ningún sistema de archivos. Los programas que se encuentran allí acceden a los módulos de GRUB en la partición de arranque. La ubicación predeterminada es */boot/grub/*.

La ubicación de la partición de arranque es una decisión del usuario que afecta a la configuración. Se recomienda tener una partición pequeña (de 200 MB de tamaño sugerido) solo para la información de arranque. De esta forma, cada compilación, ya sea LFS o alguna distribución comercial, puede acceder a los mismos archivos de arranque desde cualquier sistema arrancado. Si decide hacerlo, deberá montar

la partición independiente y mover todos los archivos del directorio */boot* actual (por ejemplo, el kernel de Linux que acaba de compilar en la sección anterior) a la nueva partición. A continuación, deberá desmontar la partición y volver a montarla como */boot*. Si lo hace, asegúrese de actualizar */etc/fstab*.

Dejar */boot* en la partición LFS actual también funcionará, pero la configuración para varios sistemas es más compleja.

Con la información anterior, determine el designador adecuado para la partición raíz (o la partición de arranque, si se utiliza una independiente). Para el siguiente ejemplo, se asume que la partición raíz (o de arranque independiente) es `sda2`.

Instale los archivos de GRUB en */boot/grub* y configure la ruta de arranque:

Advertencia

El siguiente comando sobrescribirá el gestor de arranque actual. No lo ejecute si no lo desea, por ejemplo, si utiliza un gestor de arranque externo para gestionar el Registro Maestro de Arranque (MBR).

```
grub-install /dev/sda
```

Nota

Si el sistema se ha iniciado mediante UEFI, **grub-install** intentará instalar los archivos para el destino *x86_64-efi*, pero estos archivos no se instalaron en el Capítulo 8. En ese caso, añada *--target i386-pc* al comando anterior.

10.4.4. Creación del archivo de configuración de GRUB

Generar /`boot/grub/grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod part_gpt
insmod ext2
set root=(hd0,2)
set gfxpayload=1024x768x32

menuentry "GNU/Linux, Linux 6.13.4-lfs-12.3" {
        linux   /boot/vmlinuz-6.13.4-lfs-12.3 root=/dev/sda2 ro
}
EOF
```

Los comandos **insmod** cargan los módulos de GRUB llamados *part_gpt* y *ext2*. A pesar del nombre, *ext2* admite los sistemas de archivos *ext2*, *ext3* y *ext4*. El comando grub-install ha incrustado algunos módulos en la imagen principal de GRUB (instalada en el MBR o en la partición BIOS de GRUB) para acceder a los demás módulos (en */boot/grub/i386-pc*) sin problemas. Por lo tanto, con una configuración típica, estos dos módulos ya están incrustados y esos dos comandos **insmod** no harán nada. Sin embargo, no son perjudiciales y pueden ser necesarios en algunas configuraciones poco comunes.

El comando **set gfxpayload=1024x768x32** establece la resolución y la profundidad de color del framebuffer VESA que se pasará al kernel. Es necesario que el controlador SimpleDRM del kernel utilice el framebuffer VESA. Puede usar un valor de resolución o profundidad de color diferente que se adapte mejor a su monitor.

Nota

Desde la perspectiva de GRUB, los archivos del kernel son relativos a la partición utilizada. Si utilizó una partición /boot independiente, elimine /boot de la línea de Linux anterior. También necesitarás cambiar la línea *root* establecida para que apunte a la partición de arranque.

Nota

El designador GRUB de una partición puede cambiar si agregó o quitó discos (incluidos discos extraíbles como memorias USB). Este cambio puede provocar un fallo de arranque, ya que *grub.cfg* hace referencia a designadores antiguos. Para evitar este problema, puede usar el UUID de una partición y el UUID de un sistema de archivos en lugar del designador GRUB para especificar un dispositivo. Ejecute `**lsblk -o UUID,PARTUUID,PATH,MOUNTPOINT**` para mostrar los UUID de sus sistemas de archivos (en la columna *UUID*) y particiones (en la columna *PARTUUID*). A continuación, reemplace `*set root=(hdx,y)*` por `*search --set=root --fs-uuid <UUID del sistema de archivos donde está instalado el kernel>*`, y reemplace `*root=/dev/sda2*` por `*root=PARTUUID=<UUID de la partición donde se compila LFS>*`.

Tenga en cuenta que el UUID de una partición es completamente diferente del UUID del sistema de archivos de dicha partición. Algunos recursos en línea podrían indicarle que use *root=UUID=<UUID del sistema de archivos>* en lugar de *root=PARTUUID=<UUID de la partición>*, pero esto requerirá un initramfs, que queda fuera del alcance de LFS.

El nombre del nodo de dispositivo de una partición en */dev* también puede cambiar (esto es menos probable que un cambio en el designador de GRUB). También puede reemplazar las rutas a nodos de dispositivo como */dev/sda1* con *PARTUUID=<UUID de la partición>*, en */etc/fstab*, para evitar un posible fallo de arranque en caso de que el nombre del nodo de dispositivo haya cambiado.

GRUB es un programa extremadamente potente que ofrece una gran cantidad de opciones para arrancar desde una amplia variedad de dispositivos, sistemas operativos y tipos de partición. También hay muchas opciones de personalización, como pantallas de inicio gráficas, reproducción de sonidos, entrada del ratón, etc. Los detalles de estas opciones quedan fuera del alcance de esta introducción.

Precaución

Existe un comando, grub-mkconfig, que puede escribir un archivo de configuración automáticamente. Utiliza un conjunto de scripts en /etc/grub.d/ y eliminará cualquier personalización que realice. Estos scripts están diseñados principalmente para distribuciones sin código fuente y no se recomiendan para LFS. Si instala una distribución comercial de Linux, es muy probable que se ejecute este programa. Asegúrese de hacer una copia de seguridad de su archivo grub.cfg.

Capítulo 11. Fin

11.1. Fin

¡Bien hecho! ¡El nuevo sistema LFS está instalado! Le deseamos mucho éxito con su nuevo y reluciente sistema Linux personalizado.

Puede ser una buena idea crear un archivo *`/etc/lfs-release`*. Con este archivo, será muy fácil para usted (y para nosotros si necesita ayuda en algún momento) averiguar qué versión de LFS está instalada en el sistema. Cree este archivo ejecutando:

```
echo 12.3 > /etc/lfs-release
```

Dos archivos que describen el sistema instalado pueden ser utilizados por paquetes que se instalen posteriormente, ya sea en formato binario o compilándolos.

El primero muestra el estado de su nuevo sistema con respecto a la Base de Estándares de Linux (LSB). Para crear este archivo, ejecute:

```
cat > /etc/lsb-release << "EOF"
DISTRIB_ID="Linux From Scratch"
DISTRIB_RELEASE="12.3"
DISTRIB_CODENAME="<su nombre aquí>"
DISTRIB_DESCRIPTION="Linux From Scratch"
EOF
```

El segundo contiene prácticamente la misma información y lo utilizan systemd y algunos entornos gráficos de escritorio. Para crear este archivo, ejecuta:

```
cat > /etc/os-release << "EOF"
NAME="Linux From Scratch"
VERSION="12.3"
ID=lfs
PRETTY_NAME="Linux From Scratch 12.3"
VERSION_CODENAME="<tu nombre aquí>"
HOME_URL="https://www.linuxfromscratch.org/lfs/"
RELEASE_TYPE="stable"
EOF
```

Asegúrate de personalizar los campos 'DISTRIB_CODENAME' y 'VERSION_CODENAME' para que el sistema sea exclusivamente tuyo.

11.2. Hazte contabilizado

Ahora que has terminado el libro, ¿quieres que te contabilicen como usuario de LFS? Visita https://www.linuxfromscratch.org/cgi-bin/lfscounter.php y regístrate como usuario de LFS introduciendo tu nombre y la primera versión de LFS que hayas usado.

Reiniciemos LFS.

11.3. Reinicio del sistema

Una vez instalado todo el software, es hora de reiniciar el ordenador. Sin embargo, aún quedan algunas cosas por comprobar. Aquí tienes algunas sugerencias:

• Si el controlador del kernel de tu hardware requiere archivos de *firmware* para funcionar correctamente, instala el `firmware` necesario.

• Asegúrate de que la contraseña del usuario *`root`* esté configurada.

• También es apropiado en este punto revisar los siguientes archivos de configuración.

- /etc/bashrc
- /etc/dircolors
- /etc/fstab
- /etc/hosts
- /etc/inputrc
- /etc/profile
- /etc/resolv.conf
- /etc/vimrc
- /root/.bash_profile
- /root/.bashrc
- /etc/sysconfig/ifconfig.eth0

Dicho esto, ¡pasemos a arrancar nuestra nueva instalación de LFS por primera vez! Primera salida del entorno *chroot*:

```
logout
```

Luego, desmonte los sistemas de archivos virtuales:

```
umount -v $LFS/dev/pts
mountpoint -q $LFS/dev/shm && umount -v $LFS/dev/shm
umount -v $LFS/dev
umount -v $LFS/run
umount -v $LFS/proc
umount -v $LFS/sys
```

Si se crearon varias particiones, desmonte las demás antes de desmontar la principal, de la siguiente manera:

```
umount -v $LFS/home
umount -v $LFS
```

Desmonte el sistema de archivos LFS:

```
umount -v $LFS
```

Ahora, reinicie el sistema.

Suponiendo que el gestor de arranque GRUB se configuró como se indicó anteriormente, el menú está configurado para arrancar *LFS 12.3* automáticamente. Al finalizar el reinicio, el sistema LFS estará listo para usarse. Verá un simple mensaje de inicio de sesión. En este punto, puede acceder al *Manual de BLFS*, donde podrá agregar más software según sus necesidades.

Si el reinicio **no** es exitoso, es hora de solucionar el problema. Para obtener consejos sobre cómo resolver problemas de arranque inicial, consulte:

https://www.linuxfromscratch.org/lfs/troubleshooting.html.

11.4. Recursos adicionales

Gracias por leer este manual de LFS. Esperamos que le haya resultado útil y que haya aprendido más sobre el proceso de creación del sistema.

Ahora que el sistema LFS está instalado, quizás se pregunte "¿Qué sigue?". Para responder a esta pregunta, hemos recopilado una lista de recursos.

- Mantenimiento

  Se informan regularmente errores y avisos de seguridad para todo el software. Dado que un sistema LFS se compila desde el código fuente, es su responsabilidad mantenerse al tanto de dichos informes. Existen varios recursos en línea que dan seguimiento a estos informes, algunos de los cuales se muestran a continuación:

  • *Avisos de Seguridad de LFS*

  Esta es una lista de vulnerabilidades de seguridad descubiertas en el libro de LFS después de su publicación.

  • *Lista de Correo de Seguridad de Código Abierto*

  Esta es una lista de correo para debatir sobre fallos de seguridad, conceptos y prácticas en la comunidad de código abierto.

- Consejos de LFS

  Los Consejos de LFS son una colección de documentos educativos enviados por voluntarios de la comunidad LFS. Los consejos están disponibles en:

  https://www.linuxfromscratch.org/hints/downloads/files/.

- Listas de Correo

  Hay varias listas de correo de LFS a las que puede suscribirse si necesita ayuda, desea mantenerse al día con los últimos desarrollos, desea contribuir al proyecto y más. Consulte el Capítulo 1 - Listas de correo para obtener más información.

- El Proyecto de Documentación de Linux

  El objetivo del Proyecto de Documentación de Linux (TLDP) es colaborar en todos los temas relacionados con la documentación de Linux. El TLDP cuenta con una amplia colección de tutoriales, guías y páginas de manual. Se encuentra en https://tldp.org/.

11.5. Primeros pasos después de LFS

11.5.1. Decidir qué hacer a continuación

Ahora que LFS está completo y tiene un sistema arrancable, ¿qué hace? El siguiente paso es decidir cómo usarlo. Generalmente, existen dos categorías generales a considerar: estación de trabajo o servidor. De hecho, estas categorías no son mutuamente excluyentes. Las aplicaciones necesarias para cada categoría se pueden combinar en un solo sistema, pero veámoslas por separado por ahora.

Un servidor es la categoría más simple. Generalmente, consiste en un servidor web como Apache *HTTP Server* y un servidor de bases de datos como MariaDB. Sin embargo, son posibles otros servicios. El sistema operativo integrado en un dispositivo de un solo uso entra en esta categoría.

Por otro lado, una estación de trabajo es mucho más compleja. Generalmente requiere un entorno gráfico de usuario como *LXDE, XFCE, KDE o Gnome*, basado en un entorno gráfico básico, y varias aplicaciones gráficas como el navegador web *Firefox*, el cliente de correo electrónico *Thunderbird* o la suite ofimática *LibreOffice*. Estas aplicaciones requieren muchos más paquetes de aplicaciones y bibliotecas de soporte (varios cientos, según las capacidades deseadas).

Además de lo anterior, existe un conjunto de aplicaciones para la administración del sistema para todo tipo de sistemas. Todas estas aplicaciones se encuentran en el manual de BLFS. No todos los paquetes son necesarios en todos los entornos. Por ejemplo, *dhcpcd* no suele ser apropiado para un servidor, y wireless_tools solo suele ser útil para un sistema portátil.

11.5.2. Trabajando en un entorno LFS básico

Al iniciar LFS por primera vez, se dispone de todas las herramientas internas para crear paquetes adicionales. Desafortunadamente, el entorno de usuario es bastante escaso. Hay un par de maneras de mejorar esto.

11.5.2.1. Trabajar desde el host LFS en chroot

Este método proporciona un entorno gráfico completo con un navegador completo y funciones de copiar y pegar. Este método permite usar aplicaciones como la versión de wget del host para descargar el código fuente de los paquetes a una ubicación disponible al trabajar en el entorno chroot.

Para compilar paquetes correctamente en chroot, también deberá recordar montar los sistemas de archivos virtuales si aún no lo están. Una forma de hacerlo es crear un script en el sistema **HOST**:

```
cat > ~/mount-virt.sh << "EOF"
#!/bin/bash

function mountbind
{
   if ! mountpoint $LFS/$1 >/dev/null; then
     $SUDO mount --bind /$1 $LFS/$1
     echo $LFS/$1 mounted
   else
     echo $LFS/$1 already mounted
```

```
   fi
}

function mounttype
{
   if ! mountpoint $LFS/$1 >/dev/null; then
     $SUDO mount -t $2 $3 $4 $5 $LFS/$1
     echo $LFS/$1 mounted
   else
     echo $LFS/$1 already mounted
   fi
}

if [ $EUID -ne 0 ]; then
  SUDO=sudo
else
  SUDO=""
fi

if [ x$LFS == x ]; then
  echo "LFS not set"
  exit 1
fi

mountbind dev
mounttype dev/pts devpts devpts -o gid=5,mode=620
mounttype proc    proc   proc
mounttype sys     sysfs  sysfs
mounttype run     tmpfs  run
if [ -h $LFS/dev/shm ]; then
  install -v -d -m 1777 $LFS$(realpath /dev/shm)
else
  mounttype dev/shm tmpfs tmpfs -o nosuid,nodev
fi

#mountbind usr/src
#mountbind boot
#mountbind home
EOF
```

Tenga en cuenta que los tres últimos comandos del script están comentados. Son útiles si esos directorios están montados como particiones separadas en el sistema host y se montarán al arrancar el sistema LFS/BLFS completo.

El script se puede ejecutar con **bash ~/mount-virt.sh** como usuario normal (recomendado) o como `root`. Si se ejecuta como usuario normal, se requiere sudo en el sistema host.

Otro problema que señala el script es dónde almacenar los archivos de los paquetes descargados. Esta ubicación es arbitraria. Puede ser el directorio de inicio de un usuario normal, como ~/sources, o una ubicación global como /usr/src. Nuestra recomendación es no mezclar fuentes BLFS y LFS en /sources (desde el entorno chroot). En cualquier caso, los paquetes deben ser accesibles dentro del entorno chroot.

Una última característica práctica que se presenta aquí es agilizar el proceso de acceso al entorno chroot. Esto se puede hacer con un alias en el archivo ~/.bashrc del usuario en el sistema host:

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

Este alias es un poco complejo debido a las comillas y los niveles de barra invertida. Debe estar todo en una sola línea. El comando anterior se ha dividido en dos para facilitar la presentación.

11.5.2.2. Trabajo remoto mediante ssh

Este método también proporciona un entorno gráfico completo, pero primero requiere instalar sshd en el sistema LFS, generalmente en chroot. También requiere un segundo ordenador. Este método tiene la ventaja de ser sencillo al no requerir la complejidad del entorno chroot. También utiliza el kernel compilado de LFS para todos los paquetes adicionales y sigue proporcionando un sistema completo para la instalación de paquetes.

Puede usar el comando **scp** para cargar el código fuente de los paquetes que se compilarán en el sistema LFS. Si prefiere descargar el código fuente directamente en el sistema LFS, instale *libtasn1*, *p11-kit*, *make-ca* y *wget* en chroot (o cargue sus fuentes mediante **scp** después de iniciar el sistema LFS).

11.5.2.3. Trabajar desde la línea de comandos de LFS

Este método requiere instalar libtasn1, p11-kit, make-ca, wget, gpm y links (o lynx) en chroot y luego reiniciar el nuevo sistema **LFS**. En este punto, el sistema predeterminado tiene seis consolas virtuales. Cambiar de consola es tan fácil como usar la combinación de teclas **Alt+Fx**, donde **Fx** está entre **F1** y **F6**. Las combinaciones **Alt+**← y **Alt+**→ también cambiarán la consola. En este punto, puede iniciar sesión en dos consolas virtuales diferentes y ejecutar los enlaces o el navegador Lynx en una consola y bash en la otra. GPM permite copiar comandos del navegador con el botón izquierdo del ratón, cambiar de consola y pegarlos en la otra.

Nota

Como nota al margen, también se puede cambiar de consola virtual desde una instancia de X Window con la combinación de teclas **Ctrl+Alt+Fx**, pero la operación de copiar con el ratón no funciona entre la interfaz gráfica y una consola virtual. Puede volver a la pantalla de X Window con la combinación **Ctrl+Alt+Fx**, donde **Fx** suele ser **F1**, pero también puede ser **F7**.