Manual de referencia de GNU Guix
Uso del gestor de paquetes funcional GNU Guix.

Las desarrolladoras de GNU Guix
Se garantiza el permiso de copia, distribución y/o modificación de este documento bajo los términos de la licencia de documentación libre de GNU (GNU Free Documentation License), versión 1.3 o cualquier versión posterior publicada por la Free Software Foundation; sin secciones invariantes, sin textos de cubierta delantera ni trasera. Una copia de la licencia está incluida en la sección titulada “GNU Free Documentation License”.
Índice General

GNU Guix ......................................................... 1

1 Introducción ............................................... 2
  1.1 Gestión de software con Guix ....................... 2
  1.2 Distribución GNU ...................................... 3

2 Instalación ................................................. 5
  2.1 Instalación binaria ..................................... 5
  2.2 Requisitos .............................................. 7
  2.3 Ejecución de la batería de pruebas ................. 8
  2.4 Preparación del daemon ............................. 9
    2.4.1 Configuración del entorno de construcción .... 9
    2.4.2 Uso de la facilidad de descarga de trabajo ... 11
    2.4.3 Soporte de SELinux .............................. 14
      2.4.3.1 Instalación de la política de SELinux .... 14
      2.4.3.2 Limitaciones ................................ 14
  2.5 Invocación de guix-daemon .......................... 15
  2.6 Configuración de la aplicación .................... 19
    2.6.1 Localizaciones .................................. 19
    2.6.2 Selector de servicios de nombres ............... 20
    2.6.3 Tipografías X11 ................................... 21
    2.6.4 Certificados X.509 .............................. 21
    2.6.5 Paquetes Emacs ................................... 21
    2.6.6 La cadena de herramientas de GCC .......... 22
  2.7 Actualizar Guix ........................................ 22

3 Instalación del sistema ................................. 23
  3.1 Limitaciones ........................................... 23
  3.2 Consideraciones sobre el hardware ................. 23
  3.3 Instalación desde memoria USB y DVD ............... 24
    Copiado en una memoria USB .......................... 24
    Grabación en un DVD .................................. 24
    Arranque ................................................. 25
  3.4 Preparación para la instalación ..................... 25
  3.5 Instalación gráfica guiada .......................... 25
  3.6 Instalación manual .................................. 28
    3.6.1 Distribución de teclado, red y particionado .. 28
      3.6.1.1 Distribución de teclado .................... 29
      3.6.1.2 Red ........................................ 29
      3.6.1.3 Particionado de discos .................... 30
    3.6.2 Procedimiento de instalación ................... 31
  3.7 Tras la instalación del sistema .................... 33
3.8 Instalación de Guix en una máquina virtual .................. 33
3.9 Construcción de la imagen de instalación ...................... 34
3.10 Construcción de la imagen de instalación para placas ARM ... 34

4 Gestión de paquetes ........................................... 35
4.1 Características .................................................. 35
4.2 Invocación de guix package ................................... 36
4.3 Sustituciones .................................................. 44
  4.3.1 Servidor oficial de sustituciones .......................... 44
  4.3.2 Autorización de servidores de sustituciones .......... 45
  4.3.3 Verificación de sustituciones ............................. 46
  4.3.4 Configuración de la pasarela ......................... 46
  4.3.5 Fallos en las sustituciones ............................... 46
  4.3.6 Sobre la confianza en binarios ....................... 47
4.4 Paquetes con múltiples salidas ................................ 47
4.5 Invocación de guix gc ........................................ 48
4.6 Invocación de guix pull ...................................... 51
4.7 Canales ...................................................... 54
  4.7.1 Uso de un canal de Guix personalizado ................ 55
  4.7.2 Especificación de canales adicionales ................ 55
  4.7.3 Declaración de dependencias de canales ........... 56
  4.7.4 Módulos de paquetes en un subdirectorio ........... 57
  4.7.5 Escritura de noticias del canal ....................... 57
  4.7.6 Replicación de Guix ................................... 58
4.8 Invocación de guix time-machine ........................... 59
4.9 Inferiores .................................................. 60
4.10 Invocación de guix describe ................................ 62
4.11 Invocación de guix archive ................................. 63

5 Desarrollo ....................................................... 66
5.1 Invocación de guix environment .............................. 66
5.2 Invocación de guix pack ..................................... 71

6 Interfaz programática .......................................... 76
6.1 Módulos de paquetes .......................................... 76
6.2 Definición de paquetes ...................................... 77
  6.2.1 Referencia de package .................................. 81
  6.2.2 Referencia de origin .................................... 83
6.3 Sistemas de construcción .................................... 85
6.4 El almacén .................................................. 98
6.5 Derivaciones ................................................ 101
6.6 La mónada del almacén ..................................... 104
6.7 Expresiones-G ............................................... 108
6.8 Invocación de guix repl ..................................... 117
7 Utilidades ........................................ 119
    7.1 Invocación de guix build .......................... 119
    7.1.1 Opciones comunes de construcción ................ 119
    7.1.2 Opciones de transformación de paquetes .......... 122
    7.1.3 Opciones de construcción adicionales .......... 124
    7.1.4 Depuración de fallos de construcción .......... 127
    7.2 Invocación de guix edit ............................ 129
    7.3 Invocación de guix download ......................... 129
    7.4 Invocación de guix hash ........................... 130
    7.5 Invocación de guix import ......................... 131
    7.6 Invocación de guix refresh ......................... 136
    7.7 Invocación de guix lint ............................. 141
    7.8 Invocación de guix size ............................ 143
    7.9 Invocación de guix graph ........................... 145
    7.10 Invocación de guix publish ......................... 149
    7.11 Invocación de guix challenge ....................... 153
    7.12 Invocación de guix copy ........................... 155
    7.13 Invocación de guix container ....................... 156
    7.14 Invocación de guix weather ......................... 157
    7.15 Invocación de guix processes ....................... 159

8 Configuración del sistema ............................. 161
    8.1 Uso de la configuración del sistema .................. 161
        Cargador de arranque .................................. 162
        Paquetes visibles globalmente ....................... 163
        Servicios del sistema ................................ 163
        Instanciación del sistema ............................ 167
        La interfaz programática .............................. 168
    8.2 Referencia de operating-system ....................... 168
    8.3 Sistemas de ficheros ................................. 172
    8.4 Dispositivos traducidos .............................. 175
    8.5 Cuentas de usauria .................................. 176
    8.6 Distribución de teclado .............................. 179
    8.7 Localizaciones ....................................... 181
        8.7.1 Consideraciones sobre la compatibilidad de datos de localización .... 182
    8.8 Servicios ............................................ 183
        8.8.1 Servicios base .................................... 183
        8.8.2 Ejecución de tareas programadas ................ 196
        8.8.3 Rotación del registro de mensajes ................ 198
        8.8.4 Servicios de red .................................. 200
        8.8.5 Sistema X Window ................................ 218
        8.8.6 Servicios de impresión ............................ 224
        8.8.7 Servicios de escritorio ........................... 237
        8.8.8 Servicios de sonido ............................... 245
        8.8.9 Servicios de bases de datos ...................... 247
        8.8.10 Servicios de correo .............................. 249
        8.8.11 Servicios de mensajería ......................... 276
8.8.12 Servicios de telefonía .................................................. 285
8.8.13 Servicios de monitorización ............................................. 288
8.8.14 Servicios Kerberos ....................................................... 295
8.8.15 Servicios LDAP ............................................................ 297
8.8.16 Servicios Web ............................................................. 303
8.8.17 Servicios de certificados ................................................ 320
8.8.18 Servicios DNS ............................................................. 323
8.8.19 Servicios VPN ............................................................. 335
8.8.20 Sistema de ficheros en red ............................................ 339
8.8.21 Integración continua ..................................................... 342
8.8.22 Servicios de gestión de energía ....................................... 344
8.8.23 Servicios de audio ......................................................... 352
8.8.24 Servicios de virtualización ............................................. 353
8.8.25 Servicios de control de versiones ..................................... 364
8.8.26 Servicios de juegos ....................................................... 380
8.8.27 Servicio PAM Mount ..................................................... 380
8.8.28 Servicios de Guix ......................................................... 382
8.8.29 Servicio Early OOM ...................................................... 383
8.8.30 Servicios misceláneos .................................................... 384
8.8.31 Servicios de diccionario ................................................ 386
8.9 Programas con setuid .......................................................... 389
8.10 Certificados X.509 .......................................................... 390
8.11 Selector de servicios de nombres ...................................... 391
8.12 Disco en RAM inicial ....................................................... 393
8.13 Configuración del gestor de arranque ................................. 396
8.14 Invocación de guix system ............................................... 399
8.15 Invocación de guix deploy ............................................... 407
8.16 Ejecución de Guix en una máquina virtual ......................... 410
  8.16.1 Conexión a través de SSH .......................................... 411
  8.16.2 Uso de virt-viewer con Spice ...................................... 412
8.17 Definición de servicios .................................................... 412
  8.17.1 Composición de servicios ........................................... 412
  8.17.2 Tipos de servicios y servicios .................................... 414
  8.17.3 Referencia de servicios ............................................. 416
  8.17.4 Servicios de Shepherd .............................................. 420

9 Documentación .............................................................. 423

10 Instalación de ficheros de depuración ................................. 424

11 Actualizaciones de seguridad ........................................... 425
12 Lanzamiento inicial ........................................ 427
   12.1 El lanzamiento inicial a partir de la semilla binaria reducida . 427
   12.2 Preparación para usar los binarios del lanzamiento inicial . . 429
Construcción de las herramientas de construcción ................. 430
Construir los binarios de lanzamiento ................................ 432
Reducción del conjunto de binarios de lanzamiento .................. 432

13 Transportar a una nueva plataforma ........................... 434

14 Contribuir ...................................................... 435
   14.1 Construcción desde Git ............................................. 435
   14.2 Ejecución de Guix antes de estar instalado ......................... 436
   14.3 La configuración perfecta ............................................ 437
   14.4 Guías de empaquetamiento ........................................... 438
      14.4.1 Libertad del software .......................................... 439
      14.4.2 Nombrado de paquetes ......................................... 440
      14.4.3 Versiones numéricas ............................................. 440
      14.4.4 Sinopsis y descripciones ......................................... 442
      14.4.5 Módulos Python .................................................. 443
         14.4.5.1 Especificación de dependencias .......................... 443
      14.4.6 Módulos Perl ..................................................... 444
      14.4.7 Paquetes Java .................................................... 444
      14.4.8 Rust Crates ...................................................... 444
      14.4.9 Tipografías ....................................................... 445
   14.5 Estilo de codificación .............................................. 445
      14.5.1 Paradigma de programación .................................. 445
      14.5.2 Módulos ............................................................ 446
      14.5.3 Tipos de datos y reconocimiento de patrones ............... 446
      14.5.4 Formato del código .............................................. 446
   14.6 Envío de parches .................................................. 447
      Envío de una serie de parches ....................................... 450
   14.7 Seguimiento de errores y parches ................................. 450
   14.8 Acceso al repositorio .............................................. 450

15 Reconocimientos ................................................. 453

Apéndice A Licencia de documentación libre GNU ................. 454

Índice de conceptos ............................................... 462

Índice programático ............................................ 469
GNU Guix

Este documento describe GNU Guix versión 1.0.1, una herramienta funcional de gestión de paquetes escrita para el sistema GNU.


Este manual se está traducido prácticamente al completo al castellano, pero es posible que se ocasionalmente algún fragmento sin traducir aquí y allá, debido a modificaciones al texto original en inglés del manual. Si encuentra fallos en esta traducción, o simplemente quiere colaborar en su evolución y mejora, le rogamos que nos contacte a través de la información de contacto del equipo de traducción (https://translationproject.org/team/es.html).
1 Introducción

GNU Guix\textsuperscript{1} es una herramienta de gestión de paquetes y una distribución del sistema GNU. Guix facilita a usuarias sin privilegios la instalación, actualización o borrado de paquetes de software, la vuelta a un conjunto de paquetes previo atómicamente, la construcción de paquetes desde las fuentes, y ayuda de forma general en la creación y mantenimiento de entornos software.

Puede instalar GNU Guix sobre un sistema GNU/Linux existente, donde complementará las herramientas disponibles sin interferencias (véase Capítulo 2 [Instalación], página 5), o puede usarse como un sistema operativo en sí mismo, el sistema Guix\textsuperscript{2}. Véase Sección 1.2 [Distribución GNU], página 3.

1.1 Gestión de software con Guix

Guix proporciona una interfaz de gestión de paquetes de línea de órdenes (véase Capítulo 4 [Gestión de paquetes], página 35), un conjunto de utilidades de línea de órdenes (véase Capítulo 7 [Utilidades], página 119), así como interfaces programáticas Scheme (véase Capítulo 6 [Interfaz programática], página 76). Su daemon de construcción es responsable de la construcción de paquetes en delegación de las usuarias (véase Sección 2.4 [Preparación del daemon], página 9) y de la descarga de binarios preconstruidos de fuentes autorizadas (véase Sección 4.3 [Sustituciones], página 44)

Guix incluye definiciones de paquetes para muchos paquetes GNU y no-GNU, todos los cuales respetan la libertad de computación de la usuaria (https://www.gnu.org/philosophy/free-sw.html). Es extensible: las usuarias pueden escribir sus propias definiciones de paquetes (véase Sección 6.2 [Definición de paquetes], página 77) y hacerlas disponibles como módulos independientes de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76). También es personalizable: las usuarias pueden derivar definiciones de paquetes especializadas de las existentes, inclusive desde la línea de órdenes (véase Sección 7.1.2 [Opciones de transformación de paquetes], página 122).

En su implementación, Guix utiliza la disciplina de gestión de paquetes funcional en la que Nix fue pionero (véase Capítulo 15 [Reconocimientos], página 453). En Guix, el proceso de construcción e instalación es visto como una función, en el sentido matemático. Dicha función toma entradas, como los guiones de construcción, un compilador, unas bibliotecas y devuelve el paquete instalado. Como función pura, su resultado únicamente depende de sus entradas—por ejemplo, no puede hacer referencia a software o guiones que no fuesen pasados explícitamente como entrada. Una función de construcción siempre produce el mismo resultado cuando se le proporciona un conjunto de entradas dado. No puede modificar el entorno del sistema que la ejecuta de ninguna forma; por ejemplo, no puede crear, modificar o borrar archivos fuera de sus directorios de construcción e instalación. Esto se consigue ejecutando los procesos de construcción en entornos aislados (o contenedores), donde únicamente sus entradas explícitas son visibles.

\textsuperscript{1} “Guix” se pronuncia tal y como se escribe en castellano, “iks” en el alfabeto fonético internacional (IPA).

\textsuperscript{2} Solíamos referirnos al sistema Guix como “Distribución de sistema Guix” o “GuixSD”. Ahora consideramos que tiene más sentido agrupar todo bajo la etiqueta “Guix” ya que, después de todo, el sistema Guix está inmediatamente disponible a través de la orden \texttt{guix system}, incluso cuando usa una distribución distinta por debajo!
El resultado de las funciones de construcción de paquetes es almacenado en la caché en el sistema de ficheros, en un directorio especial llamado el almacén (véaseSección 6.4 [El almacén], página 98). Cada paquete se instala en un directorio propio en el almacén—por defecto, bajo /gnu/store. El nombre del directorio contiene el hash de todas las entradas usadas para construir el paquete; por tanto, cambiar una entrada resulta en un nombre de directorio distinto.

Esta aproximación es el cimiento de las avanzadas características de Guix: capacidad para la actualización transaccional y vuelta-atrás de paquetes, instalación en el ámbito de la usuaria y recolección de basura de paquetes (véase Sección 4.1 [Características], página 35).

1.2 Distribución GNU

Guix viene con una distribución del sistema GNU consistente en su totalidad de software libre³. La distribución puede instalarse independientemente (véase Capítulo 3 [Instalación del sistema], página 23), pero también es posible instalar Guix como un gestor de paquetes sobre un sistema GNU/Linux existente (véase Capítulo 2 [Instalación], página 5). Para distinguir entre las dos opciones, nos referimos a la distribución independiente como el sistema Guix.

La distribución proporciona paquetes principales de GNU como GNU libc, GCC y Binutils, así como muchas aplicaciones GNU y no-GNU. La lista completa de paquetes disponibles se puede explorar en línea (https://www.gnu.org/software/guix/packages) o ejecutando guix package (véase Sección 4.2 [Invocación de guix package], página 36):

   guix package --list-available

Nuestro objetivo es proporcionar una distribución práctica con 100% software libre basada en Linux y otras variantes de GNU, con un enfoque en la promoción y la alta integridad de componentes GNU, y un énfasis en programas y herramientas que ayuden a las usuarias a ejercitar esa libertad.

Actualmente hay paquetes disponibles para las siguientes plataformas:

- **x86_64-linux**
  arquitectura x86_64 de Intel/AMD, núcleo Linux-Libre;
- **i686-linux**
  arquitectura de 32-bits Intel (IA32), núcleo Linux-Libre;
- **armhf-linux**
  arquitectura ARMv7-A con coma flotante hardware, Thumb-2 y NEON, usando la interfaz binaria de aplicaciones (ABI) EABI con coma flotante hardware, y el núcleo Linux-Libre.
- **aarch64-linux**
  procesadores ARMv8-A de 64 bits little-endian, con el núcleo Linux-Libre.
- **mips64el-linux**
  procesadores MIPS 64-bits little-endian, específicamente las series Loongson, n32 ABI, y núcleo Linux-Libre. Esta configuración no se soporta completamen-

---

³ El término “libre” aquí se refiere a la libertad proporcionada a las usuarias de dicho software (https://www.gnu.org/philosophy/free-sw.html).
te; en particular las granjas de construcción del proyecto ya no proporcionan sustituciones para esta arquitectura.

Con el sistema Guix, *declara* todos los aspectos de la configuración del sistema y Guix se hace cargo de instanciar la configuración de manera transaccional, reproducible y sin estado global (véase Capítulo 8 [Configuración del sistema], página 161). El sistema Guix usa el núcleo Linux-libre, el sistema de inicialización Shepherd (véase Sección “Introducción” en *The GNU Shepherd Manual*), las conocidas utilidades y herramientas de compilación GNU, así como el entorno gráfico o servicios del sistema de su elección.

El sistema Guix está disponible en todas las plataformas previas excepto mips64el-linux.

Para información sobre el transporte a otras arquitecturas o núcleos, véase Capítulo 13 [Transportar], página 434.

La construcción de esta distribución es un esfuerzo cooperativo, ¡y esta invitada a unirse! Véase Capítulo 14 [Contribuir], página 435, para información sobre cómo puede ayudar.
2 Instalación

**Nota:** Recomendamos el uso de este guión de shell de instalación (https://git.savannah.gnu.org/cgit/guix.git/plain/etc/guix-install.sh) para instalar Guix sobre un sistema GNU/Linux en ejecución, de aquí en adelante referido como una distribución distinta.¹ El guión automatiza la descarga, instalación y configuración inicial de Guix. Debe ejecutarse como la usuaria de administración root.

Cuando está instalado sobre una distribución distinta, GNU Guix complementa las herramientas disponibles sin interferencias. Sus datos radican exclusivamente en dos directorios, normalmente /gnu/store y /var/guix; otros ficheros en su sistema, como /etc, permanecen intactos.

Una vez instalado, Guix puede ser actualizado ejecutando guix pull (véase Sección 4.6 [Invocación de guix pull], página 51.

Si prefiere realizar los pasos de instalación manualmente o desea personalizarlos, puede encontrar útiles las siguientes instrucciones. Describen los requisitos de software de Guix, así como su instalación manual y la preparación para su uso.

2.1 Instalación binaria

Esta sección describe cómo instalar Guix en un sistema arbitrario desde un archivador autocontenido que proporciona los binarios para Guix y todas sus dependencias. Esto es normalmente más rápido que una instalación desde las fuentes, la cual es descrita en las siguientes secciones. El único requisito es tener GNU tar y Xz.

**Nota:** Le recomendamos el uso de este guión de instalación para el shell (https://git.savannah.gnu.org/cgit/guix.git/plain/etc/guix-install.sh). Automatiza la descarga, instalación y los pasos iniciales de configuración descritos a continuación. Debe ser ejecutarse como root.

La instalación consiste más o menos en los siguientes pasos:

1. Descargue el archivador con los binarios de `https://ftp.gnu.org/gnu/guix/guix-binary-1.0.1.sistema.tar.xz`, donde sistema es x86_64-linux para una máquina x86_64 que execute el núcleo Linux, etcétera.

   Asegúrese de descargar el fichero .sig asociado y de verificar la autenticidad del archivador con él, más o menos así:

   ```
   $ wget https://ftp.gnu.org/gnu/guix/guix-binary-1.0.1.sistema.tar.xz.sig
   $ gpg --verify guix-binary-1.0.1.sistema.tar.xz.sig
   ```

   Si la orden falla porque no dispone de la clave pública necesaria, entonces ejecute esta otra orden para importarla:

   ```
   $ wget https://sv.gnu.org/people/viewgpg.php?user_id=15145 \
   -qO - | gpg --import - \
   y vuelva a ejecutar la orden gpg --verify.
   ```

¹ Esta sección está dedicada a la instalación del gestor de paquetes, que puede realizarse sobre un sistema GNU/Linux ya en ejecución. Si, en vez de eso, desea instalar el sistema operativo GNU completo, véase Capítulo 3 [Instalación del sistema], página 23.
Tenga en cuenta que un aviso del tipo “Esta clave no está certificada con una firma de confianza” es normal.

2. Ahora necesita convertirse en la usuaria root. Dependiendo de su distribución, puede que tenga que ejecutar su -o sudo -i. Como root, ejecute:

```
# cd /tmp
# tar --warning=no-timestamp -xf \\
   /ruta/de/guix-binary-1.0.1.sistema.tar.xz
# mv var/guix /var/ && mv gnu /
```

Esto crea /gnu/store (véase Sección 6.4 [El almacén], página 98) y /var/guix. El último contiene un perfil listo para usar para root (vea el siguiente paso).

No extraiga el archivador en un sistema Guix ya funcionando ya que sobreescribiría sus propios ficheros esenciales.

La opción --warning=no-timestamp asegura que GNU tar no emite avisos sobre “marcas de tiempo imposibles” (dichos avisos eran emitidos por GNU tar 1.26 y anteriores; las versiones recientes están bien). Parten del hecho de que todos los ficheros en el archivador tienen su tiempo de modificación fijado a cero (que significa el 1 de enero de 1970). Esto es hecho voluntariamente para asegurarse de que el contenido del archivador es independiente de su fecha de creación, por tanto haciendo que sea reproducible.

3. Ponga disponible el perfil en ~root/.config/guix/current, que es donde guix pull instalará las actualizaciones (véase Sección 4.6 [Invocación de guix pull], página 51):

```
# mkdir -p ~root/.config/guix
# ln -sf /var/guix/profiles/per-user/root/current-guix \\
   ~root/.config/guix/current
```

Cargue etc/profile para aumentar PATH y otras variables de entorno relevantes:

```
# GUIX_PROFILE="'echo ~root/.config/guix/current" ; \\
source $GUIX_PROFILE/etc/profile
```

4. Cree el grupo y las cuentas de usuaria para las usuarias de construcción como se explica a continuación (véase Sección 2.4.1 [Configuración del entorno de construcción], página 9).

5. Ejecute el daemon, y configure su inicio automático durante el arranque.

Si su distribución anfitriona usa el sistema de inicio systemd, puede hacerlo con las siguientes órdenes:

```
# cp ~root/.config/guix/current/lib/systemd/system/guix-daemon.service \\
   /etc/systemd/system/
# systemctl start guix-daemon && systemctl enable guix-daemon
```

Si su distribución anfitriona usa el sistema de inicio Upstart:

```
# initctl reload-configuration
# cp ~root/.config/guix/current/lib/upstart/system/guix-daemon.conf \\
   /etc/init/
# start guix-daemon
```

En otro caso, todavía puede iniciar el daemon manualmente con:

```
# ~root/.config/guix/current/bin/guix-daemon \\
   --build-users-group=guixbuild
```
6. Haga accesible la orden guix a otras usuarias de la máquina, por ejemplo con:

```
# mkdir -p /usr/local/bin
# cd /usr/local/bin
# ln -s /var/guix/profiles/per-user/root/current-guix/bin/guix
```

Es también una buena idea poner disponible la versión Info de este manual ahí:

```
# mkdir -p /usr/local/share/info
# cd /usr/local/share/info
# for i in /var/guix/profiles/per-user/root/current-guix/share/info/* ;
do ln -s $i ; done
```

De este modo, asumiendo que `/usr/local/share/info` está en la ruta de búsqueda, ejecutar `info guix.es` abrirá este manual (véase Sección “Other Info Directories” en GNU Texinfo, para más detalles sobre cómo cambiar la ruta de búsqueda de Info).

7. Para usar sustituciones de `ci.guix.gnu.org` o uno de sus espejos (véase Sección 4.3 [Sustituciones], página 44), debe autorizarlas:

```
# guix archive --authorize < 
~root/.config/guix/current/share/guix/ci.guix.gnu.org.pub
```

8. Cada usuaria puede necesitar dar algunos pasos adicionales para preparar su entorno de Guix para el uso diario, véase Sección 2.6 [Configuración de la aplicación], página 19.

Voilà, ¡la instalación está completa!

Puede confirmar que Guix está funcionando instalando un paquete de ejemplo en su perfil de root:

```
# guix install hello
```

El archivador de la instalación binaria puede ser (re)producido y verificado simplemente ejecutando la siguiente orden en el árbol de fuentes de Guix:

```
make guix-binary.sistema.tar.xz
```

... que a su vez ejecuta:

```
guix pack -s sistema --localstatedir \
--profile-name=current-guix guix
```

Véase Sección 5.2 [Invocación de guix pack], página 71, para más información sobre esta útil herramienta.

## 2.2 Requisitos

Esta sección enumera los requisitos para construir Guix desde las fuentes. El procedimiento de construcción de Guix es el mismo que el de otro software GNU, y no está cubierto aquí. Por favor, eche un vistazo a los ficheros README y INSTALL en el árbol de fuentes de Guix para obtener detalles adicionales.


GNU Guix depende de los siguientes paquetes:

- GNU Guile ([https://gnu.org/software/guile/](https://gnu.org/software/guile/)), versión 3.0.x o 2.2.x;
- Guile-Gcrypt ([https://notabug.org/cwebber/guile-gcrypt](https://notabug.org/cwebber/guile-gcrypt)), versión 0.1.0 o posterior;
Las siguientes dependencias son opcionales:
- Support for build offloading (véase Sección 2.4.2 [Configuración de delegación del daemon], página 11) and guix copy (véase Sección 7.12 [Invocación de guix copy], página 155) depends on Guile-SSH (https://github.com/artyom-poptsov/guile-ssh), version 0.12.0 or later.
- Cuando lzlib (https://www.nongnu.org/lzzip/lzlib.html) está disponible, se pueden usar sustituciones con lzlib y guix publish puede comprimir las sustituciones con lzlib.
- Cuando libbz2 (http://www.bzip.org) está disponible, guix daemon puede usarla para comprimir los registros de construcción.

A menos que se pasase --disable-daemon a configure, los siguientes paquetes también son necesarios:
- GNU libgcrypt (https://gnupg.org/);
- SQLite 3 (https://sqlite.org);
- g++ de GCC (https://gcc.gnu.org) con soporte para el estándar C++11

Cuando se configura Guix en un sistema que ya tiene una instalación de Guix, asegúrese de especificar el mismo directorio de estado que el de la instalación existente usando la opción --localstatedir al guión configure (véase Sección “Directory Variables” en GNU Coding Standards). Habitualmente, esta opción localstatedir tiene el valor /var. El guión configure le proteje ante una mala configuración no deseada de localstatedir de modo que no pueda corromper inadvertidamente su almacén (véase Sección 6.4 [El almacén], página 98).

### 2.3 Ejecución de la batería de pruebas

Después de una ejecución exitosa de configure y make, es una buena idea ejecutar la batería de pruebas. Puede ayudar a encontrar problemas con la configuración o el entorno, o errores en el mismo Guix—e informar de fallos en las pruebas es realmente una buena forma de ayudar a mejorar el software. Para ejecutar la batería de pruebas, teclee:
```
make check
```
Los casos de prueba pueden ejecutarse en paralelo: puede usar la opción -j de GNU make para acelerar las cosas. La primera ejecución puede tomar algunos minutos en una máquina reciente; las siguientes ejecuciones serán más rápidas puesto que el almacén creado para las pruebas ya tendrá varias cosas en la caché.
También es posible ejecutar un subconjunto de las pruebas definiendo la variable de makefile `TESTS` como en el ejemplo:

```
make check TESTS="tests/store.scm tests/cpio.scm"
```

Por defecto, los resultados de las pruebas se muestran a nivel de fichero. Para ver los detalles de cada caso de prueba individual, es posible definir la variable de makefile `SCM_LOG_DRIVER_FLAGS` como en el ejemplo:

```
make check TESTS="tests/base64.scm" SCM_LOG_DRIVER_FLAGS="--brief=no"
```

En caso de fallo, le rogamos que envíe un correo a `bug-guix@gnu.org` y adjunte el fichero `test-suite.log`. Por favor, especifique la versión de Guix usada así como los números de versión de las dependencias (véase Sección 2.2 [Requisitos], página 7) en su mensaje.

Guix también viene como una batería de pruebas del sistema completo que prueban instancias completas del sistema Guix. Se puede ejecutar únicamente en sistemas donde Guix ya está instalado, usando:

```
make check-system
```

o, de nuevo, definiendo `TESTS` para seleccionar un subconjunto de las pruebas a ejecutar:

```
make check-system TESTS="basic mcron"
```

Estas pruebas de sistema están definidas en los módulos (`gnu tests ...`). Funcionan ejecutando el sistema operativo con una instrumentación ligera en una máquina virtual (VM). Pueden ser computacionalmente intensivas o bastante baratas, dependiendo de si hay sustituciones disponibles para sus dependencias (véase Sección 4.3 [Sustituciones], página 44). Algunas requieren mucho espacio de almacenamiento para alojar las imágenes de la máquina virtual.

De nuevo, en caso de fallos en las pruebas, le rogamos que envíe a `bug-guix@gnu.org` todos los detalles.

### 2.4 Preparación del daemon

Operaciones como la construcción de un paquete o la ejecución del recolector de basura son realizadas por un proceso especializado, el *daemon de construcción*, en delegación de sus clientes. Únicamente el daemon puede acceder al almacén y su base de datos asociada. Por tanto, cualquier operación que manipule el almacén se realiza a través del daemon. Por ejemplo, las herramientas de línea de órdenes como `guix package` y `guix build` se comunican con el daemon (vía llamadas a procedimientos remotos) para indicarle qué hacer.

Las siguientes secciones explican cómo preparar el entorno del daemon de construcción. Véase también Sección 4.3 [Sustituciones], página 44, para información sobre cómo permitir al daemon descargar binarios pre-construidos.

#### 2.4.1 Configuración del entorno de construcción

En una configuración multiusuario estándar, Guix y su daemon—el programa `guix-daemon`—son instalados por la administradora del sistema; `/gnu/store` pertenece a `root` y `guix-daemon` se ejecuta como `root`. Usuarias sin privilegios pueden usar las herramientas de Guix para construir paquetes o acceder al almacén de otro modo, y el daemon lo hará en delegación suya, asegurando que el almacén permanece en un estado consistente, y permitiendo compartir entre usuarias los paquetes construidos.
Mientras que el guix-daemon se ejecuta como root, puede que no desee que los procesos de construcción de paquetes se ejecuten como root también, por razones de seguridad obvias. Para evitarlo, una reserva especial de 'usuarios de construcción' debe ser creada para ser usada por los procesos de construcción iniciados por el daemon. Estas 'usuarios de construcción' no necesitan tener un shell ni un directorio home: simplemente serán usadas cuando el daemon se deshaga de los privilegios de root en los procesos de construcción. Tener varias de dichas 'usuarios de construcción' permite al daemon lanzar distintos procesos de construcción bajo UID separados, lo que garantiza que no interferirán entre ellos—una característica esencial ya que las construcciones se caracterizan como funciones puras (véase Capítulo 1 [Introducción], página 2).

En un sistema GNU/Linux, una reserva de 'usuarios de construcción' puede ser creada así (usando la sintaxis de Bash y las órdenes de shadow):

```
# groupadd --system guixbuild
# for i in `seq -w 1 10`;
do
    useradd -g guixbuild -G guixbuild -d /var/empty -s `which nologin` -c "Usuaria de construcción Guix $i" --system guixbuilder$i;
done
```

El número de 'usuarios de construcción' determina cuantos trabajos de construcción se pueden ejecutar en paralelo, especificado por la opción --max-jobs (véase Sección 2.5 [Invocación de guix-daemon], página 15). Para usar guix system vm y las órdenes relacionadas, puede necesitar añadir las 'usuarios de construcción' al grupo kvm para que puedan acceder a /dev/kvm, usando -G guixbuild,kvm en vez de -G guixbuild (véase Sección 8.14 [Invocación de guix system], página 399).

El programa guix-daemon puede ser ejecutado entonces como root con la siguiente orden:

```
# guix-daemon --build-users-group=guixbuild
```

De este modo, el daemon inicia los procesos de construcción en un "chroot", bajo una de las 'usuarios de construcción'. En GNU/Linux, por defecto, el entorno "chroot" contiene únicamente:

- un directorio /dev mínimo, creado en su mayor parte independientemente del /dev del sistema anfitrión;
- el directorio /proc; únicamente muestra los procesos del contenedor ya que se usa un espacio de nombres de PID separado;
- /etc/passwd con una entrada para la 'usuario' actual y una entrada para la 'usuario' nobody;
- /etc/groups con una entrada para el grupo de la 'usuario';

2 Si su máquina usa el sistema de inicio systemd, copiando el fichero prefix/lib/systemd/system/guix-daemon.service en /etc/systemd/system asegurará que el guix-daemon se arranca automáticamente. De igual modo, si su máquina usa el sistema de inicio Upstart, copie el fichero prefix/lib/upstart/system/guix-daemon.conf en /etc/init.

3 "En su mayor parte" porque, mientras el conjunto de ficheros que aparecen en /dev es fijo, la mayor parte de estos ficheros solo pueden ser creados si el sistema anfitrión los tiene.
• /etc/hosts con una entrada que asocia localhost a 127.0.0.1;
• un directorio /tmp con permisos de escritura.

Puede influir en el directorio que el daemon utiliza para almacenar los árboles de construcción *via* la variable de entorno TMPDIR. No obstante, el árbol de construcción en el “chroot” siempre se llama /tmp/guix-build-nombre.drv-0, donde nombre es el nombre de la derivación—por ejemplo, coreutils-8.24. De este modo, el valor de TMPDIR no se escapa a los entornos de construcción, lo que evita discrepancias en caso de que los procesos de construcción capturen el nombre de su árbol de construcción.

El daemon también respeta la variable de entorno http_proxy para las descargas HTTP que realiza, sea para derivaciones de salida fija (véase Sección 6.5 [Derivaciones], página 101) o para sustituciones (véase Sección 4.3 [Sustituciones], página 44).

Si está instalando Guix como una usuaria sin privilegios, es posible todavía ejecutar guix-daemon siempre que pase --disable-chroot. No obstante, los procesos de construcción no estarán aislados entre sí ni del resto del sistema. Por tanto, los procesos de construcción pueden interferir entre ellos y pueden acceder a programas, bibliotecas y otros ficheros disponibles en el sistema—haciendo mucho más difícil verlos como funciones puras.

### 2.4.2 Uso de la facilidad de descarga de trabajo

Cuando así se desee, el daemon de construcción puede delegar construcciones de derivación a otras máquinas ejecutando Guix, usando el procedimiento de extensión de construcción offload4. Cuando dicha característica es activada, una lista de máquinas de construcción especificadas por la usuaria es leída de /etc/guix/machines.scm; cada vez que se solicita una construcción, por ejemplo guix build, el daemon intenta delegarla a una de las máquinas que satisfaga las condiciones de la derivación, en particular su tipo de sistema—por ejemplo, x86_64-linux. Los prerrequisitos restantes para la construcción son copiados por SSH a la máquina objetivo, la cual procede con la construcción; con un resultado satisfactorio la(s) salida(s) de la construcción son copiadas de vuelta a la máquina inicial.

El fichero /etc/guix/machines.scm normalmente tiene un contenido de este estilo:

```scheme
(list (build-machine
       (name "ochentayseis.example.org")
       (system "x86_64-linux")
       (host-key "ssh-ed25519 AAAAC3Nza...")
       (user "rober")
       (speed 2.)) ;¡increíblemente rápida!

(build-machine
       (name "mimips.example.org")
       (system "mips64el-linux")
       (host-key "ssh-rsa AAAAB3Nza...")
       (user "alicia")
       (private-key
        (string-append (getenv "HOME")
                       "/.ssh/identidad-para-guix"))))
```

---

En el ejemplo anterior se especifica una lista de dos máquinas de construcción, una para la arquitectura x86_64 y otra para la arquitectura mips64el. De hecho, este fichero es—¡sin sorpresa ninguna!—un fichero Scheme que se evalúa cuando el procedimiento de extensión offload se inicia. El valor que devuelve debe ser una lista de objetos build-machine. Mientras que este ejemplo muestra una lista fija de máquinas de construcción, una puede imaginarse, digamos, el uso de DNS-SD para devolver una lista de máquinas de construcción potenciales descubierta en la red local (véase Sección “Introducción” en Using Avahi in Guile Scheme Programs). El tipo de datos build-machine se detalla a continuación.

build-machine
Este tipo de datos representa las máquinas de construcción a las cuales el daemon puede delegar construcciones. Los campos importantes son:

- **name**: El nombre de red de la máquina remota.
- **system**: El sistema de la máquina remota—por ejemplo, "x86_64-linux".
- **user**: La cuenta de usuario usada para la conexión a la máquina remota por SSH. Tenga en cuenta que el par de claves SSH no debe estar protegido por contraseña, para permitir ingresos al sistema no interactivos.
- **host-key**: Este campo debe contener la clave pública de la máquina de SSH en formato OpenSSH. Es usado para autenticar la máquina cuando nos conectamos a ella. Es una cadena larga más o menos así:
  
  `ssh-ed25519 AAAAC3NzaC...mde+UhL recordatorio@example.org`

  Si la máquina está ejecutando el daemon OpenSSH, sshd, la clave pública de la máquina puede encontrarse en un fichero como `/etc/ssh/ssh_host_ed25519_key.pub`.

  Si la máquina está ejecutando el daemon SSH GNU lsh, lshd, la clave de la máquina está en `/etc/lsh/host-key.pub` o un fichero similar. Puede convertirse a formato OpenSSH usando `lsh-export-key` (véase Sección “Converting keys” en LSH Manual):

  `$ lsh-export-key --openssh < /etc/lsh/host-key.pub
  ssh-rsa AAAAB3NzaC1yc2EAAAAE0p8FoQAAAQEAs1eB46LV...`

Ciertos número de campos opcionales pueden ser especificados:

- **port** (predeterminado: 22)
  Número de puerto del servidor SSH en la máquina.

- **private-key** (predeterminado: `~root/.ssh/id_rsa`)
  El fichero de clave privada SSH usado para conectarse a la máquina, en formato OpenSSH. Esta clave no debe estar protegida con una contraseña. Tenga en cuenta que el valor predeterminado es la clave privada de la cuenta de root. Asegúrese de que existe si usa el valor predeterminado.

- **compression** (predeterminado: "zlib@openssh.com,zlib")
- **compression-level** (predeterminado: 3)
  Los métodos de compresión y nivel de compresión a nivel SSH solicitados.
Tenga en cuenta que la delegación de carga depende de la compresión SSH para reducir el ancho de banda usado cuando se transfieren ficheros hacia y desde máquinas de construcción.

**daemon-socket** (predeterminado: "/var/guix/daemon-socket/socket")
Nombre de fichero del socket de dominio Unix en el que guix-daemon escucha en esa máquina.

**parallel-builds** (predeterminadas: 1)
El número de construcciones que pueden ejecutarse en paralelo en la máquina.

**speed** (predeterminado: 1.0)
Un “factor de velocidad relativa”. El planificador de delegaciones tenderá a preferir máquinas con un factor de velocidad mayor.

**features** (predeterminadas: `()`) 
Una lista de cadenas denotando las características específicas permitidas por la máquina. Un ejemplo es "kvm" para máquinas que tienen los módulos KVM de Linux y las correspondientes características hardware. Las derivaciones pueden solicitar las características por nombre, y entonces se planificarán en las máquinas adecuadas.

El ejecutable guix debe estar en la ruta de búsqueda de las máquinas de construcción. Puede comprobar si es el caso ejecutando:

```bash
ssh build-machine guix repl --version
```

Hay una última cosa por hacer una vez `machines.scm` está en su lugar. Como se ha explicado anteriormente, cuando se delega, los ficheros se transfieren en ambas direcciones entre los almacenes de las máquinas. Para que esto funcione, primero debe generar un par de claves en cada máquina para permitir al daemon exportar los archivos firmados de ficheros en el almacén (véase Sección 4.11 [Invocación de guix archive], página 63):

```bash
# guix archive --generate-key
```

Cada máquina de construcción debe autorizar a la clave de la máquina maestra para que acepte elementos del almacén que reciba de la maestra:

```bash
# guix archive --authorize < clave-publica-maestra.txt
```

Del mismo modo, la máquina maestra debe autorizar la clave de cada máquina de construcción.

Todo este lío con claves está ahí para expresar las mutuas relaciones de confianza entre pares de la máquina maestra y las máquinas de construcción. Concretamente, cuando la maestra recibe ficheros de una máquina de construcción (y vice versa), su daemon de construcción puede asegurarse de que son genuinos, no han sido modificados, y que están firmados por una clave autorizada.

Para comprobar si su configuración es operacional, ejecute esta orden en el nodo maestro:

```bash
# guix offload test
```

 Esto intentará conectar con cada una de las máquinas de construcción especificadas en `/etc/guix/machines.scm`, comprobará que GUile y los módulos Guix están disponibles en cada máquina, intentará exportar a la máquina e importar de ella, e informará de cualquier error en el proceso.
Si quiere probar un fichero de máquinas diferente, simplemente lo debe especificar en la línea de órdenes:

```
# guix offload test otras-maquinas.scm
```

Por último, puede probar un subconjunto de máquinas cuyos nombres coincidan con una expresión regular así:

```
# guix offload test maquinas.scm ‘\.gnu\.org$’
```

Para mostrar la carga actual de todas las máquinas de construcción, ejecute esta orden en el nodo principal:

```
# guix offload status
```

### 2.4.3 Soporte de SELinux

Guix incluye un fichero de política SELinux en `etc/guix-daemon.cil` que puede ser instalado en un sistema donde SELinux está activado, para etiquetar los ficheros Guix y especificar el comportamiento esperado del daemon. Ya que el sistema Guix no proporciona una política base de SELinux, la política del daemon no puede usarse en el sistema Guix.

#### 2.4.3.1 Instalación de la política de SELinux

Para instalar la política ejecute esta orden como root:

```
semodule -i etc/guix-daemon.cil
```

Una vez hecho, vuelva a etiquetar el sistema de ficheros con `restorecon` o con un mecanismo distinto que proporcione su sistema.

Una vez la política está instalada, el sistema de ficheros ha sido re-etiquetado, y el daemon ha sido reiniciado, debería ejecutarse en el contexto `guix_daemon_t`. Puede confirmarlo con la siguiente orden:

```
ps -Zax | grep guix-daemon
```

Monitorice los ficheros de log de SELinux mientras ejecuta una orden como `guix build hello` para convencerse que SELinux permite todas las operaciones necesarias.

#### 2.4.3.2 Limitaciones

Esta política no es perfecta. Aquí está una lista de limitaciones o comportamientos extraños que deben ser considerados al desplegar la política SELinux provista para el daemon Guix.

1. `guix_daemon_socket_t` no se usa realmente. Ninguna de las operaciones del socket implica contextos que tengan algo que ver con `guix_daemon_socket_t`. No hace daño tener esta etiqueta sin usar, pero sería preferible definir reglas del socket únicamente para esta etiqueta.

2. `guix gc` no puede acceder enlaces arbitrarios a los perfiles. Por diseño, la etiqueta del fichero del destino de un enlace simbólico es independiente de la etiqueta de fichero del fichero en sí. Aunque todos los perfiles bajo `$localstatedir` se etiquetan, los enlaces para estos perfiles heredan la etiqueta del directorio en el que están. Para enlaces en el directorio de la usuaria esto será `user_home_t`. Pero para los enlaces del directorio de root, o `/tmp`, o del directorio del servidor HTTP, etc., esto no funcionará. `guix gc` se verá incapacitado para leer y seguir dichos enlaces.
3. La característica del daemon de esperar conexiones TCP puede que no funcione más. Esto puede requerir reglas adicionales, ya que SELinux trata los sockets de red de forma diferente a los ficheros.

4. Actualmente todos los ficheros con un nombre coincidente con la expresión regular /gnu/store.+-(gux-.+|profile)/bin/guix-daemon tienen asignada la etiqueta guix_daemon_exec_t; esto significa que cualquier fichero con ese nombre en cualquier perfil tendrá permitida la ejecución en el dominio guix_daemon_t. Esto no es ideal. Una atacante podría construir un paquete que proporcione este ejecutable y convencer a la usuaria para instalarlo y ejecutarlo, lo que lo eleva al dominio guix_daemon_t. Llegadas a este punto, SELinux no puede prevenir que acceda a los ficheros permitidos para los procesos en dicho dominio.

Podríamos generar una política mucho más restrictiva en tiempo de instalación, de modo que solo el nombre exacto del fichero del ejecutable de guix-daemon actualmente instalado sea marcado como guix_daemon_exec_t, en vez de usar una expresión regular amplia. La desventaja es que root tendría que instalar o actualizar la política en tiempo de instalación cada vez que se actualizase el paquete de Guix que proporcione el ejecutable de guix-daemon realmente en ejecución.

2.5 Invocación de guix-daemon

El programa guix-daemon implementa toda la funcionalidad para acceder al almacen. Esto incluye iniciar procesos de construcción, ejecutar el recolector de basura, comprobar la disponibilidad de un resultado de construcción, etc. Normalmente se ejecuta como root así:

```
# guix-daemon --build-users-group=guixbuild
```

Para detalles sobre cómo configurarlo, véase Sección 2.4 [Preparación del daemon], página 9.

Por defecto, guix-daemon inicia los procesos de construcción bajo distintos UIDs, tomados del grupo de construcción especificado con --build-users-group. Además, cada proceso de construcción se ejecuta en un entorno "chroot" que únicamente contiene el subconjunto del almacen del que depende el proceso de construcción, como especifica su derivación (véase Capítulo 6 [Interfaz programática], página 76), más un conjunto específico de directorios del sistema. Por defecto, estos directorios contienen /dev y /dev/pts. Es más, sobre GNU/Linux, el entorno de construcción es un contenedor; además de tener su propio árbol del sistema de ficheros, tiene un espacio de nombres de montado separado, su propio espacio de nombres de PID, de red, etc. Esto ayuda a obtener construcciones reproducibles (véase Sección 4.1 [Características], página 35).

Cuando el daemon realiza una construcción en delegación de la usuaria, crea un directorio de construcción bajo /tmp o bajo el directorio especificado por su variable de entorno TMPDIR. Este directorio se comparte con el contenedor durante toda la construcción, aunque dentro del contenedor el árbol de construcción siempre se llama /tmp/guix-build-nombre.drv-0.

El directorio de construcción se borra automáticamente una vez completado el proceso, a menos que la construcción fallase y se especificase en el cliente --keep-failed (véase Sección 7.1 [Invocación de guix build], página 119).

El daemon espera conexiones y lanza un subproceso por sesión iniciada por cada cliente (una de las sub-órdenes de guix). La orden guix processes le permite tener una visión ge-
Capítulo 2: Instalación

neral de la actividad de su sistema mostrando clientes y sesiones activas. Véase Sección 7.15 [Invocación de guix processes], página 159, para más información.

Se aceptan las siguientes opciones de línea de ordenes:

--build-users-group=grupo
Toma las usuarias de grupo para ejecutar los procesos de construcción (véase Sección 2.4 [Preparación del daemon], página 9).

--no-substitutes
No usa sustituciones para la construcción de productos. Esto es, siempre realiza las construcciones localmente en vez de permitir la descarga de binarios preconstruidos (véase Sección 4.3 [Sustituciones], página 44).

Cuando el daemon se ejecuta con --no-substitutes, los clientes aún pueden activar explícitamente las sustituciones via la llamada de procedimiento remoto set-build-options (véase Sección 6.4 [El almacén], página 98).

--substitute-urls=urls
Considera urls la lista separada por espacios predeterminada de URLs de sustituciones de fuentes. Cuando se omite esta opción, se usa ‘https://ci.guix.gnu.org’.

 Esto significa que las sustituciones puede ser descargadas de urls, mientras estén firmadas por una firma de confianza (véase Sección 4.3 [Sustituciones], página 44).

--no-offload
No usa la delegación de construcciones en otras máquinas (véase Sección 2.4.2 [Configuración de delegación del daemon], página 11). Es decir, siempre realiza las construcciones de manera local en vez de delegar construcciones a máquinas remotas.

--cache-failures
Almacena en la caché los fallos de construcción. Por defecto, únicamente las construcciones satisfactorias son almacenadas en la caché.

Cuando se usa esta opción, guix gc --list-failures puede usarse para consultar el conjunto de elementos del almacén marcados como fallidos; guix gc --clear-failures borra los elementos del almacén del conjunto de fallos existentes en la caché. Véase Sección 4.5 [Invocación de guix gc], página 48.

--cores=n
-c n
Usa n núcleos de la CPU para construir cada derivación; 0 significa tantos como haya disponibles.

El valor predeterminado es 0, pero puede ser sobreescrito por los clientes, como la opción --cores de guix build (véase Sección 7.1 [Invocación de guix build], página 119).

El efecto es definir la variable de entorno NIX_BUILD_CORES en el proceso de construcción, el cual puede usarla para explotar el paralelismo interno—por ejemplo, ejecutando make -j$NIX_BUILD_CORES.
--max-jobs=n

Permite como máximo $n$ trabajos de construcción en paralelo. El valor predeterminado es 1. Fijarlo a 0 significa que ninguna construcción se realizará localmente; en vez de eso, el daemon delegará las construcciones (véase Sección 2.4.2 [Configuración de delegación del daemon], página 11), o simplemente fallará.

--max-silent-time=segundos

Cuando la construcción o sustitución permanece en silencio más de segundos, la finaliza e informa de un fallo de construcción.

El valor predeterminado es 0, que desactiva los plazos.

El valor especificado aquí puede ser sobreescrito por clientes (véase Sección 7.1.1 [Opciones comunes de construcción], página 119).

--timeout=segundos

Del mismo modo, cuando el proceso de construcción o sustitución dura más de segundos, lo termina e informa un fallo de construcción.

El valor predeterminado es 0, que desactiva los plazos.

El valor especificado aquí puede ser sobreescrito por los clientes (véase Sección 7.1.1 [Opciones comunes de construcción], página 119).

--rounds=N

Construye cada derivación $n$ veces seguidas, y lanza un error si los resultados de las construcciones consecutivas no son idénticos bit-a-bit. Fíjese que esta configuración puede ser sobreescrita por clientes como guix build (véase Sección 7.1 [Invocación de guix build], página 119).

Cuando se usa conjuntamente con --keep-failed, la salida que difiere se mantiene en el almacén, bajo /gnu/store/...-check. Esto hace fácil buscar diferencias entre los dos resultados.

--debug

Produce salida de depuración.

 Esto es útil para depurar problemas en el arranque del daemon, pero entonces puede ser cambiado el comportamiento por los clientes, por ejemplo la opción --verbosity de guix build (véase Sección 7.1 [Invocación de guix build], página 119).

--chroot-directory=dir

Añade dir al chroot de construcción.

Hacer esto puede cambiar el resultado del proceso de construcción—por ejemplo si usa dependencias opcionales, que se encuentren en dir, cuando están disponibles, y no de otra forma. Por esa razón, no se recomienda hacerlo. En vez de eso, asegúrese que cada derivación declara todas las entradas que necesita.

--disable-chroot

Desactiva la construcción en un entorno chroot.

No se recomienda el uso de esta opción ya que, de nuevo, podría permitir a los procesos de construcción ganar acceso a dependencias no declaradas. Es necesario, no obstante, cuando guix-daemon se ejecuta bajo una cuenta de usuaria sin privilegios.
Capítulo 2: Instalación

--log-compression=tipo
Comprime los logs de construcción de acuerdo a tipo, que puede ser gzip, bzip2 o none.
A menos que se use --lose-logs, todos los log de construcción se mantienen en localstatedir. Para ahorrar espacio, el daemon automáticamente los comprime con bzip2 por defecto.

--disable-deduplication
Desactiva la “deduplicación” automática en el almacén.
Por defecto, los ficheros se añaden al almacén “deduplicados” automáticamente: si un nuevo fichero añadido es idéntico a otro que ya se encuentra en el almacén, el daemon introduce el nuevo fichero como un enlace duro al otro fichero. Esto puede reducir notablemente el uso del disco, a expenses de una carga de entrada/salida ligeramente incrementada al finalizar un proceso de construcción. Esta opción desactiva dicha optimización.

--gc-keep-outputs=[yes|no]
Determina si el recolector de basura (GC) debe mantener salidas de las derivaciones vivas.
Cuando se usa “yes”, el recolector de basura mantendrá las salidas de cualquier derivación viva disponible en el almacén—los ficheros .drv. El valor predeterminado es “no”, lo que significa que las salidas de las derivaciones se mantienen únicamente si son alcanzables desde alguna raíz del recolector de basura. Véase Sección 4.5 [Invocación de guix gc], página 48, para más información sobre las raíces del recolector de basura.

--gc-keep-derivations=[yes|no]
Determina si el recolector de basura (GC) debe mantener derivaciones correspondientes a salidas vivas.
Cuando se usa “yes”, como es el caso predeterminado, el recolector de basura mantiene derivaciones—es decir, ficheros .drv—mientras al menos una de sus salidas está viva. Esto permite a las usuarias seguir la pista de los orígenes de los elementos en el almacén. El uso de “no” aquí ahorra un poco de espacio en disco.
De este modo, usar --gc-keep-derivations con valor “yes” provoca que la vitalidad fluya de salidas a derivaciones, y usar --gc-keep-outputs con valor “yes” provoca que la vitalidad fluya de derivaciones a salidas. Cuando ambas tienen valor “yes”, el efecto es mantener todos los prerrequisitos de construcción (las fuentes, el compilador, las bibliotecas y otras herramientas de tiempo de construcción) de los objetos vivos del almacén, independientemente de que esos prerrequisitos sean alcanzables desde una raíz del recolector de basura. Esto es conveniente para desarrolladoras ya que ahorra reconstrucciones o descargas.

--impersonate-linux-2.6
En sistemas basados en Linux, suplanta a Linux 2.6. Esto significa que la llamada del sistema uname del kernel indicará 2.6 como el número de publicación. Esto puede ser útil para construir programas que (habitualmente de forma incorrecta) dependen en el número de versión del núcleo.
--lose-logs
No guarda logs de construcción. Por defecto se almacenan bajo localstatedir/guix/log.

--system=sys
Asume sistema como el tipo actual de sistema. Por defecto es el par de arquitectura/núcleo encontrado durante la configuración, como x86_64-linux.

--listen=destino
Espera conexiones en destino. destino se interpreta como el nombre del fichero del socket de dominio Unix si comienza on / (barra a la derecha). En otro caso, destino se interpreta como un nombre de máquina o un nombre de máquina y puerto a escuchar. Aquí van unos pocos ejemplos:

--listen=/gnu/var/daemon
Espera conexiones en el socket de dominio Unix /gnu/var/daemon, se crea si es necesario.

--listen=localhost
Espera conexiones TCP en la interfaz de red correspondiente a localhost, en el puerto 44146.

--listen=128.0.0.42:1234
Espera conexiones TCP en la interfaz de red correspondiente a 128.0.0.42, en el puerto 1234.

Esta opción puede repetirse múltiples veces, en cuyo caso guix-daemon acepta conexiones en todos los destinos especificados. Las usuarias pueden indicar a los clientes a qué destino conectarse fijando la variable de entorno GUIX_DAEMON_SOCKET (véase Sección 6.4 [El almacén], página 98).

Nota: El protocolo del daemon no está autenticado ni cifrado. El uso de --listen=dirección es aceptable en redes locales, como clusters, donde únicamente los nodos de confianza pueden conectarse al daemon de construcción. En otros casos donde el acceso remoto al daemon es necesario, recomendamos usar sockets de dominio Unix junto a SSH.

Cuando se omite --listen, guix-daemon escucha conexiones en el socket de dominio Unix que se encuentra en localstatedir/guix/daemon-socket/socket.

2.6 Configuración de la aplicación

Cuando se usa Guix sobre una distribución GNU/Linux distinta al sistema Guix—una distribución distinta—unos pocos pasos adicionales son necesarios para tener todo preparado. Aquí están algunos de ellos.

2.6.1 Localizaciones

Los paquetes instalados via Guix no usarán los datos de localización del sistema anfitrión. En vez de eso, debe primero instalar uno de los paquetes de localización disponibles con Guix y después definir la variable de entorno GUIX_LOCPATH:

$ guix install glibc-locales
La variable `GUIX_LOCPATH` juega un rol similar a `LOCPATH` (véase Sección “Locale Names” en The GNU C Library Reference Manual). No obstante, hay dos diferencias importantes:

1. `GUIX_LOCPATH` es respetada únicamente por la libc dentro de Guix, y no por la libc que proporcionan las distribuciones distintas. Por tanto, usar `GUIX_LOCPATH` le permite asegurarse de que los programas de la distribución distinta no cargarán datos de localización incompatibles.

2. libc añade un sufijo a cada entrada de `GUIX_LOCPATH` con `/X.Y`, donde `X.Y` es la versión de libc—por ejemplo, `2.22`. Esto significa que, en caso de que su perfil Guix contenga una mezcla de programas enlazados contra diferentes versiones de libc, cada versión de libc únicamente intentará cargar datos de localización en el formato correcto.

Esto es importante porque el formato de datos de localización usado por diferentes versiones de libc puede ser incompatible.

### 2.6.2 Selector de servicios de nombres

Cuando se usa Guix en una distribución distinta, recomendamos encarecidamente que el sistema ejecute el daemon de caché del servicio de nombres de la biblioteca de C de GNU, `nscd`, que debe escuchar en el socket `/var/run/nscd/socket`. En caso de no hacerlo, las aplicaciones instaladas con Guix pueden fallar al buscar nombres de máquinas o cuentas de usuaria, o incluso pueden terminar abruptamente. Los siguientes párrafos explican por qué.

La biblioteca de C de GNU implementa un selector de servicios de nombres (NSS), que es un mecanismo extensible para “búsquedas de nombres” en general: resolución de nombres de máquinas, cuentas de usuaria y más (véase Sección “Selector de servicios de nombres” en The GNU C Library Reference Manual).

Al ser extensible, NSS permite el uso de módulos, los cuales proporcionan nuevas implementaciones de búsqueda de nombres: por ejemplo, el módulo `nss-mdns` permite la resolución de nombres de máquina `.local`, el módulo `nis` permite la búsqueda de cuentas de usuaria usando el servicio de información de red (NIS), etc. Estos “servicios de búsqueda” extra se configuran para todo el sistema en `/etc/nsswitch.conf`, y todos los programas en ejecución respetan esta configuración (véase Sección “NSS Configuration File” en The GNU C Reference Manual).

Cuando se realiza una búsqueda de nombres—por ejemplo, llamando a la función `getaddrinfo` en C—las aplicaciones primero intentarán conectar con `nscd`; en caso satisfactorio, `nscd` realiza la búsqueda de nombres en delegación suya. Si `nscd` no está ejecutándose, entonces realizan la búsqueda por ellas mismas, cargando los servicios de búsqueda de nombres en su propio espacio de direcciones y ejecutándola. Estos servicios de búsqueda de nombres—los ficheros `libnss_*.*`—son abiertos con `dlopen`, pero pueden venir de la biblioteca de C del sistema, en vez de la biblioteca de C contra la que la aplicación está enlazada (la biblioteca de C que viene en Guix).

Y aquí es donde está el problema: si su aplicación está enlazada contra la biblioteca de C de Guix (digamos, `glibc 2.24`) e intenta cargar módulos de otra biblioteca de C (digamos,
libnss_mdns.so para glibc 2.22), probablemente terminará abruptamente o sus búsquedas de nombres fallarán inesperadamente.

Ejecutar nscl en el sistema, entre otras ventajas, elimina este problema de incompatibilidad binaria porque esos ficheros libnss_* so se cargan en el proceso nscl, no en la aplicación misma.

### 2.6.3 Tipografías X11

La mayoría de aplicaciones gráficas usan Fontconfig para encontrar y cargar tipografías y realizar la renderización del lado del cliente X11. El paquete fontconfig en Guix busca tipografías en $HOME/.guix-profile por defecto. Por tanto, para permitir a aplicaciones gráficas instaladas con Guix mostrar tipografías, tiene que instalar las tipografías también con Guix. Paquetes esenciales de tipografías incluyen gs-fonts, font-dejavu y font-gnu-freefont-ttf.

Para mostrar texto escrito en lenguas chinas, Japonés o Coreano en aplicaciones gráficas, considere instalar font-adobe-source-han-sans o font-wqy-zenhei. La anterior tiene múltiples salidas, una por familia de lengua (véase Sección 4.4 [Paquetes con múltiples salidas], página 47). Por ejemplo, la siguiente orden instala tipografías para lenguas chinas:

```
  guix install font-adobe-source-han-sans:cn
```

Programas más antiguos como xterm no usan Fontconfig sino que dependen en el lado del servidor para realizar el renderizado de tipografías. Dichos programas requieren especificar un nombre completo de tipografía usando XLFD (Descripción lógica de tipografías X), como esta:

```
  -*-dejavu sans-medium-r-normal-*-100-*-*-*--1
```

Para ser capaz de usar estos nombres completos para las tipografías TrueType instaladas en su perfil Guix, necesita extender la ruta de fuentes del servidor X:

```
  xset +fp $(dirname $(readlink -f ~/.guix-profile/share/fonts/truetype/fonts.dir))
```

Después de eso, puede ejecutar xlsfonts (del paquete xlsfonts) para asegurarse que sus tipografías TrueType se enumeran aquí.

Después de instalar tipografías puede tener que refrescar la caché de tipografías para usarlas en las aplicaciones. Lo mismo aplica cuando las aplicaciones instaladas vía Guix no parecen encontrar tipografías. Para forzar la reconstrucción de la caché de tipografías ejecute fc-cache -rv. La orden fc-cache es proporcionada por el paquete fontconfig.

### 2.6.4 Certificados X.509

El paquete nss-certs proporciona certificados X.509, que permiten a los programas verificar los servidores accedidos por HTTPS.

Cuando se usa Guix en una distribución distinta, puede instalar este paquete y definir las variables de entorno relevantes de modo que los paquetes sepan dónde buscar los certificados. Véase Sección 8.10 [Certificados X.509], página 390, para información detallada.

### 2.6.5 Paquetes Emacs

When you install Emacs packages with Guix, the Elisp files are placed under the share/emacs/site-lisp/ directory of the profile in which they are installed. The Elisp libraries are made available to Emacs through the EMACSLOADPATH environment variable, which is set when installing Emacs itself.
Capítulo 2: Instalación

Además, las definiciones de autoload se evalúan automáticamente durante la inicialización de Emacs, por el procedimiento guix-emacs-autoload-packages específico de Guix. Si, por algún motivo, quieres evitar la auto-carga de los paquetes de Emacs instalados con Guix, puedes hacerlo al ejecutar Emacs con la opción --no-site-file (véase Sección “Init File” en The GNU Emacs Manual).

2.6.6 La cadena de herramientas de GCC

Guix ofrece paquetes de compiladores individuales como gcc, pero si necesitas una cadena completa de herramientas para compilar y enlazar código fuente lo que realmente deseas es el paquete gcc-toolchain. Este paquete proporciona una cadena de herramientas GCC para desarrollo C/C++, incluyendo el mismo GCC, la biblioteca de C GNU (cabeceras y binarios, más símbolos de desarrollo en la salida debug), Binutils y un recubrimiento del enlazador.

El propósito del recubrimiento es inspeccionar las opciones -L y -l proporcionadas al enlazador, y los correspondientes parámetros -rpath, y llamar al enlazador real con este nuevo conjunto de parámetros. Puede instruir al recubrimiento para rechazar el enlace contra bibliotecas que no se encuentren en el almacén fijando el valor de la variable de entorno GUIX_LD_WRAPPER_ALLOW_IMPURITIES a no.

2.7 Actualizar Guix

Para actualizar Guix ejecuta:

guix pull

Véase Sección 4.6 [Invocación de guix pull], página 51, para más información.

En una distribución distinta puede actualizar el daemon de construcción ejecutando:

sudo -i guix pull

seguido de (asumiendo que la su distribución usa la herramienta de gestión de servicios systemd):

systemctl restart guix-daemon.service

En el Sistema Guix, la actualización del daemon se lleva a cabo con la reconfiguración el sistema (véase Sección 8.14 [Invocación de guix system], página 399).
3 Instalación del sistema

Esta sección explica cómo instalar el sistema Guix en una máquina. Guix, como gestor de paquetes, puede instalarse sobre un sistema GNU/Linux en ejecución, véase Capítulo 2 [Instalación], página 5.

3.1 Limitaciones

Consideramos que el sistema Guix está listo para un amplio rango de casos de uso, tanto de servidor como de escritorio. Las garantías que proporciona—actualizaciones transaccionales y vuelta atrás atómica, reproducibilidad—lo convierten en un cimiento sólido.

No obstante, antes de que proceda con la instalación, sea consciente de las siguientes limitaciones apreciables que se conocen en la versión 1.0.1:

- No está implementada la funcionalidad del gestor de volúmenes lógicos (LVM).
- Se proporcionan más y más servicios del sistema (véase Sección 8.8 [Servicios], página 183), pero pueden faltar algunos.
- Están disponibles GNOME, Xfce, LXDE y Enlightenment (véase Sección 8.8.7 [Servicios de escritorio], página 237), así como un número de gestores de ventanas X11. No obstante, actualmente falta KDE.

Más que una descarga de responsabilidades es una invitación a informar de problemas (¡e historias satisfactorias!), y para unirse a nosotras en su mejora. Véase Capítulo 14 [Contribuir], página 435, para más información.

3.2 Consideraciones sobre el hardware

GNU Guix se enfoca en respetar la libertad de computación de las usuarias. Se construye sobre el núcleo Linux-libre, lo que significa que únicamente funciona hardware para el que existen controladores y firmware libres. Hoy en día, un amplio rango del hardware común funciona con GNU/Linux-libre—desde teclados a tarjetas gráficas a escáneres y controladoras Ethernet. Desafortunadamente, todavía hay áreas donde los fabricantes de hardware deniegan a las usuarias el control de su propia computación, y dicho hardware no funciona en el sistema Guix.

Una de las áreas principales donde faltan controladores o firmware libre son los dispositivos WiFi. Los dispositivos WiFi que se sabe que funcionan incluyen aquellos que usan los chips Atheros (AR9271 y AR7010), que corresponden al controlador ath9k de Linux-libre, y aquellos que usan los chips Broadcom/AirForce (BCM43xx con Wireless-Core Revisión 5), que corresponden al controlador b43-open de Linux-libre. Existe firmware libre para ambos, y está disponible por defecto en el sistema Guix, como parte de %base-firmware (véase Sección 8.2 [Referencia de operating-system], página 168).

La Fundación del Software Libre (https://www.fsf.org/) patrocina Respeta Su Libertad (https://www.fsf.org/ryf) (RYF), un programa de certificación para productos hardware que respetan su libertad y su privacidad y se aseguran de que usted tenga el control sobre su dispositivo. Le recomendamos que compruebe la lista de dispositivos certificados RYF.

Otro recurso útil es el sitio web H-Node (https://www.h-node.org/). Contiene un catálogo de dispositivos hardware con información acerca su funcionalidad con GNU/Linux.
3.3 Instalación desde memoria USB y DVD

Se puede descargar una imagen de instalación ISO-9660 que puede ser escrita en una memoria USB o grabada en un DVD desde https://ftp.gnu.org/gnu/guix/guix-system-install-1.0.1.sistema.iso.xz, donde sistema es uno de los siguientes valores:

- x86_64-linux
  para un sistema GNU/Linux en CPUs compatibles con la arquitectura de 64-bits de Intel/AMD;

- i686-linux
  para un sistema GNU/Linux en CPUs compatibles con la arquitectura de 32-bits de Intel.

Asegúrese de descargar el fichero .sig asociado y de verificar la autenticidad de la imagen contra él, más o menos así:

```
$ gpg --verify guix-system-install-1.0.1.sistema.iso.xz.sig
```

Si la orden falla porque no dispone de la clave pública necesaria, entonces ejecute esta otra orden para importarla:

```
$ wget https://sv.gnu.org/people/viewgpg.php?user_id=15145 -qO - | gpg --import -
y vuelva a ejecutar la orden gpg --verify.
```

Tenga en cuenta que un aviso del tipo “Esta clave no esta certificada con una firma de confianza” es normal.

Esta imagen contiene las herramientas necesarias para una instalación. Está pensada para ser copiada tal cual a una memoria USB o DVD con espacio suficiente.

Copiado en una memoria USB

Para copiar la imagen en una memoria USB, siga estos pasos:

1. Descomprima la imagen usando la orden xz:

   ```
   xz -d guix-system-install-1.0.1.sistema.iso.xz
   ```

2. Conecte una memoria USB de 1 GiB o más a su máquina, y determine su nombre de dispositivo. Asumiendo que la memoria USB es /dev/sdX copie la imagen con:

   ```
   dd if=guix-system-install-1.0.1.sistema.iso of=/dev/sdX sync
   ```

   El acceso a /dev/sdX normalmente necesita privilegios de root.

Grabación en un DVD

Para copiar la imagen a un DVD, siga estos pasos:

1. Descomprima la imagen usando la orden xz:

   ```
   xz -d guix-system-install-1.0.1.sistema.iso.xz
   ```

2. Introduzca un DVD en su máquina para grabarlo, y determine el nombre del dispositivo. Asumiendo que la unidad DVD es /dev/srX, copie la imagen con:

   ```
   growisofs -dvd-compat -Z /dev/srX=guix-system-install-1.0.1.sistema.iso
   ```

   El acceso a /dev/srX normalmente necesita privilegios de root.
Capítulo 3: Instalación del sistema

Arranque

Una vez hecho esto, debe ser capaz de reiniciar el sistema y arrancar desde la memoria USB o el DVD. Para lo primero habitualmente es necesario introducirse en la BIOS o en el menú de arranque UEFI, donde se puede seleccionar el arranque desde la memoria USB. Para arrancar desde Libreboot, cambie a la línea de ordenes pulsando la tecla `c` y teclee `search_grub usb`.

Véase Sección 3.8 [Instalación de Guix en una máquina virtual], página 33, si, en vez de esto, desea instalar el sistema Guix en una máquina virtual (VM).

3.4 Preparación para la instalación

Una vez que haya arrancado, puede usar el instalador gráfico guiado, el cual facilita la introducción al sistema (véase Sección 3.5 [Instalación gráfica guiada], página 25). Alternativamente, si ya es está familiarizada con GNU/Linux y desea más control que el que proporciona el instalador gráfico, puede seleccionar el proceso de instalación “manual” (véase Sección 3.6 [Instalación manual], página 28).

El instalador gráfico está disponible en TTY1. Puede obtener consolas de root en los TTY 3 a 6 pulsando `ctrl-alt-f3`, `ctrl-alt-f4`, etc. TTY2 muestra esta documentación y se puede cambiar a dicha consola con `ctrl-alt-f2`. La documentación es explorable usando las órdenes del lector Info (véase Stand-alone GNU Info). El sistema de instalación ejecuta el daemon GPM para ratones, el cual le permite seleccionar texto con el botón izquierdo y pegarlo con el botón central.

Nota: La instalación requiere acceso a Internet de modo que cualquier dependencia de su configuración de sistema no encontrada pueda ser descargada. Véase la sección “Red” más adelante.

3.5 Instalación gráfica guiada

El instalador gráfico es una interfaz de usaria basada en texto. Le guiará, con cajas de diálogo, a través de los pasos necesarios para instalar el sistema GNU Guix.

Las primeras cajas de diálogo le permiten configurar el sistema mientras lo usa durante la instalación: puede seleccionar el idioma, la distribución del teclado y configurar la red, la
cual se usará durante la instalación. La siguiente imagen muestra el diálogo de configuración de red.
Los siguientes pasos le permitirán particionar su disco duro, como se muestra en la siguiente imagen, elegir si se usarán o no sistemas de ficheros cifrados, introducir el nombre de la máquina, la contraseña de root y crear cuentas adicionales, entre otras cosas.
Tenga en cuenta que, en cualquier momento, el instalador le permite salir de la instalación actual y retomarla en un paso previo, como se muestra en la siguiente imagen.

Una vez haya finalizado, el instalador produce una configuración de sistema operativo y la muestra (véase Sección 8.1 [Uso de la configuración del sistema], página 161). En este punto puede pulsar “OK” y la instalación procederá. En caso de finalización satisfactoria, puede reiniciar con el nuevo sistema y disfrutarlo. ¡Véase Sección 3.7 [Tras la instalación del sistema], página 33, para ver cómo proceder a continuación!

3.6 Instalación manual

Esta sección describe cómo podría instalar “manualmente” el sistema GNU Guix en su máquina. Esta opción requiere familiaridad con GNU/Linux, con el shell y con las herramientas de administración comunes. Si piensa que no es para usted, considere el uso del instalador gráfico guiado (véase Sección 3.5 [Instalación gráfica guiada], página 25).

El sistema de instalación proporciona consolas de root en los terminales virtuales (TTY) 3 a 6; pulse `ctrl-alt-f3`, `ctrl-alt-f4` y sucesivas teclas para abrirlas. Incluye muchas herramientas comunes necesarias para la instalación del sistema. Pero es también un sistema Guix completo, lo que significa que puede instalar paquetes adicionales, en caso de necesitarlos, mediante el uso de `guix package` (véase Sección 4.2 [Invocación de guix package], página 36).

3.6.1 Distribución de teclado, red y particionado

Antes de instalar el sistema, puede desear ajustar la distribución del teclado, configurar la red y particionar el disco duro deseado. Esta sección le guiará durante este proceso.
3.6.1.1 Distribución de teclado
La imagen de instalación usa la distribución de teclado QWERTY de los EEUU. Si desea cambiarla, puede usar la orden `loadkeys`. Por ejemplo, la siguiente orden selecciona la distribución de teclado para el castellano:

```bash
loadkeys es
```

Véanse los ficheros bajo `/run/current-system/profile/share/keymaps` para la obtención de una lista de distribuciones de teclado disponibles. Ejecute `man loadkeys` para más información.

3.6.1.2 Red
Ejecute la siguiente orden para ver los nombres asignados a sus interfaces de red:

```bash
ifconfig -a
```

... o, usando la orden específica de GNU/Linux `ip`:

```bash
ip address
```

El nombre de las interfaces de cable comienza con ‘e’; por ejemplo, la interfaz que corresponde a la primera controladora Ethernet en la placa se llama ‘eno1’. El nombre de las interfaces inalámbricas comienza con ‘w’, como ‘w1p2s0’.

Conexión por cable
Para configurar una red por cable ejecute la siguiente orden, substituyendo `interfaz` con el nombre de la interfaz de cable que desea usar.

```bash
ifconfig interfaz up
```

... o, usando la orden específica de GNU/Linux `ip`:

```bash
ip link set interfaz up
```

Conexión sin cable
Para configurar una red inalámbrica, puede crear un fichero de configuración para la herramienta de configuración `wpa_supplicant` (su ruta no es importante) usando uno de los editores de texto disponibles como `nano`:

```bash
nano wpa_supplicant.conf
```

Como un ejemplo, la siguiente plantilla puede colocarse en este fichero y funcionará para muchas redes inalámbricas, siempre que se proporcione el SSID y la contraseña reales de la red a la que se va a conectar:

```bash
network={
    ssid="mi-ssid"
    key_mgmt=WPA-PSK
    psk="la contraseña de la red"
}
```

Inicie el servicio inalámbrico y lance su ejecución en segundo plano con la siguiente orden (sustituya `interfaz` por el nombre de la interfaz de red que desea usar):

```bash
wpa_supplicant -c wpa_supplicant.conf -i interfaz -B
```

Ejecute `man wpa_supplicant` para más información.
En este punto, necesita obtener una dirección IP. En una red donde las direcciones IP se asignan automáticamente mediante DHCP, puede ejecutar:

```
dhcclient -v interfaz
```

Intente hacer ping a un servidor para comprobar si la red está funcionando correctamente:

```
ping -c 3 gnu.org
```

Configurar el acceso por red es casi siempre un requisito debido a que la imagen no contiene todo el software y las herramientas que puedan ser necesarias.

Si lo desea, puede continuar la instalación de forma remota iniciando un servidor SSH:

```
herd start ssh-daemon
```

Asegúrese de fijar una contraseña con `passwd`, o configurar la verificación de clave pública de OpenSSH para la introducción en el sistema.

### 3.6.1.3 Particionado de discos

A menos que se haya realizado previamente, el siguiente paso es el particionado, y después dar formato a la/s partición/es deseadas.

La imagen de instalación contiene varias herramientas de particionado, incluyendo Parted (véase Sección “Overview” en GNU Parted User Manual), fdisk y cfdisk. Invoque su ejecución y configure el mapa de particiones deseado en su disco:

```
cfdisk
```

Si su disco usa el formato de tabla de particiones GUID (GPT) y tiene pensado instalar GRUB basado en BIOS (la opción predeterminada), asegúrese de tener una partición de arranque BIOS disponible (véase Sección “BIOS installation” en GNU GRUB manual).

Si en vez de eso desea GRUB basado en EFI, se requiere una Partición del Sistema EFI (ESP) con formato FAT32. Esta partición puede montarse en `/boot/efi` y debe tener la opción esp activa. Por ejemplo, en `parted`:

```
parted /dev/sda set 1 esp on
```

**Nota:** ¿No está segura si usar GRUB basado en EFI o en BIOS? Si el directorio `/sys/firmware/efi` existe en la imagen de instalación, probablemente debería realizar una instalación EFI, usando `grub-efi-bootloader`. En otro caso, debe usar GRUB basado en BIOS, conocido como `grub-bootloader`. Véase Sección 8.13 [Configuración del gestor de arranque], página 396, para más información sobre cargadores de arranque.

Una vez haya terminado con el particionado de la unidad de disco deseada, tiene que crear un sistema de ficheros en la o las particiones relevantes\(^1\). Para la partición ESP, si tiene una y asumiendo que es `/dev/sda1`, execute:

```
mkfs.fat -F32 /dev/sda1
```

Preferentemente, asigne una etiqueta a los sistemas de ficheros de modo que pueda referirse a ellos de forma fácil y precisa en las declaraciones `file-system` (véase Sección 8.3 [Sistemas de ficheros], página 172). Esto se consigue habitualmente con la opción `-L` de

---
\(^1\) Actualmente el sistema Guix únicamente permite sistemas de ficheros ext4, btrfs y JFS. En particular, el código que lee UUIDs del sistema de ficheros y etiquetas únicamente funciona para dichos sistemas de ficheros.
Capítulo 3: Instalación del sistema

mkfs.ext4 y las ordenes relacionadas. Por tanto, asumiendo que la partición de la raíz es /dev/sda2, se puede crear un sistema de ficheros con la etiqueta mi-raiz de esta manera:

    mkfs.ext4 -L mi-raiz /dev/sda2

Si en vez de eso planea cifrar la partición raíz, puede usar las herramientas Cryptsetup/LUKS para hacerlo (véase man cryptsetup para más información). Asumiendo que quiere almacenar la partición raíz en /dev/sda2, la secuencia de ordenes sería más o menos así:

    cryptsetup luksFormat /dev/sda2
    cryptsetup open --type luks /dev/sda1 mi-particion
    mkfs.ext4 -L mi-raiz /dev/mapper/mi-particion

Una vez hecho esto, monte el sistema de ficheros deseado bajo /mnt con una orden como (de nuevo, asumiendo que mi-raiz es la etiqueta del sistema de ficheros raíz):

    mount LABEL=mi-raiz /mnt

Monte también cualquier otro sistema de ficheros que desee usar en el sistema resultante relativamente a esta ruta. Si ha optado por /boot/efi como el punto de montaje de EFI, por ejemplo, ahora debe ser montada en /mnt/boot/efi para que guix system init pueda encontrarla más adelante.

Finalmente, si planea usar una o más particiones de intercambio (véase Sección “Memory Concepts” en The GNU C Library Reference Manual), asegúrese de inicializarla con mkswap. Asumiendo que tuviese una partición de intercambio en /dev/sda3, ejecutaría:

    mkswap /dev/sda3
    swapon /dev/sda3

De manera alternativa, puede usar un fichero de intercambio. Por ejemplo, asumiendo que en el nuevo sistema desea usar el fichero /fichero-de-intercambio como tal, ejecutaría2:

    # Esto son 10GiB de espacio de intercambio. Ajuste "count" para
    # cambiar el tamaño.
    dd if=/dev/zero of=/mnt/swapfile bs=1MiB count=10240
    # Por seguridad, se le conceden permisos de lectura y escritura
    # únicamente a root.
    chmod 600 /mnt/swapfile
    mkswap /mnt/swapfile
    swapon /mnt/swapfile

Fíjese que si ha cifrado la partición raíz y creado un fichero de intercambio en su sistema de ficheros como se ha descrito anteriormente, el cifrado también protege al fichero de intercambio, como a cualquier fichero en dicho sistema de ficheros.

3.6.2 Procedimiento de instalación

Con las particiones deseadas listas y la raíz deseada montada en /mnt, estamos preparadas para empezar. Primero, ejecute:

    herd start cow-store /mnt

---

2 Este ejemplo funcionará para muchos tipos de sistemas de ficheros (por ejemplo, ext4). No obstante, para los sistemas de ficheros con mecanismos de copia-durante-escritura (por ejemplo, btrfs) los pasos pueden ser diferentes. Para obtener más detalles, véanse las páginas de manual para mkswap y swapon.
Capítulo 3: Instalación del sistema

Esto activa la copia-durante-escritura en `/gnu/store`, de modo que los paquetes que se añadan durante la fase de instalación se escriban en el disco montado en `/mnt` en vez de permanecer en memoria. Esto es necesario debido a que la primera fase de la orden `guix system init` (vea más adelante) implica descargas o construcciones en `/gnu/store`, el cual, inicialmente, está un sistema de ficheros en memoria.

Después debe editar un fichero y proporcionar la declaración de sistema operativo a instalar. Para dicho fin, el sistema de instalación viene con tres editores de texto. Recomendamos GNU nano (véase GNU nano Manual), que permite el resaltado de sintaxis y correspondencia de paréntesis; los otros editores son GNU Zile (un clon de Emacs) y nvi (un clon del editor vi original de BSD). Le recomendamos encarecidamente almacenar ese fichero en el sistema de ficheros raíz, digamos, como `/mnt/etc/config.scm`. En caso de no hacerlo, habrá perdido su configuración del sistema una vez arranque en el sistema recién instalado.

Véase Sección 8.1 [Uso de la configuración del sistema], página 161, para hacerse una idea del fichero de configuración. Las configuraciones de ejemplo mencionadas en esa sección están disponibles bajo `/etc/configuration` en la imagen de instalación. Por tanto, para empezar con una configuración del sistema que proporcione un servidor gráfico (un sistema de “escritorio”), puede ejecutar algo parecido a estas órdenes:

```bash
# mkdir /mnt/etc
# cp /etc/configuration/desktop.scm /mnt/etc/config.scm
# nano /mnt/etc/config.scm
```

Debe prestar atención a lo que su fichero de configuración contiene, y en particular:

- Asegúrese que la forma `bootloader-configuration` especifica la localización deseada de la instalación de GRUB. Debe mencionar `grub-bootloader` si está usando GRUB con el arranque antiguo, o `grub-efi-bootloader` para sistemas más nuevos UEFI. Para los sistemas antiguos, el campo `target` denomina un dispositivo, como `/dev/sda`; para los sistemas UEFI denomina la ruta de una partición EFI montada, como `/boot/efi`; asegúrese de que la ruta está actualmente montada y haya una entrada `file-system` especificada en su configuración.

- Asegúrese que las etiquetas de su sistema de ficheros corresponden con el valor de sus campos `device` respectivos en su configuración `file-system`, asumiendo que su configuración `file-system` usa el procedimiento `file-system-label` en su campo `device`.

- Si hay particiones cifradas o en RAID, asegúrese de añadir un campo `mapped-devices` para describirlas (véase Sección 8.4 [Dispositivos traducidos], página 175).

Una vez haya terminado de preparar el fichero de configuración, el nuevo sistema debe ser inicializado (recuerde que el sistema de ficheros raíz deseado está montado bajo `/mnt`):

```
# guix system init /mnt/etc/config.scm /mnt
```

Esto copia todos los ficheros necesarios e instala GRUB en `/dev/sdX`, a menos que proporcione la opción `--no-bootloader`. Para más información, véase Sección 8.14 [Invocación de guix system], página 399. Esta orden puede desencadenar descargas o construcciones de paquetes no encontrados, lo cual puede tomar algún tiempo.

Una vez que la orden se complete—¡y, deseablemente, de forma satisfactoria!—puede ejecutar `reboot` y arrancar con el nuevo sistema. La contraseña de `root` en el nuevo sistema…
Capítulo 3: Instalación del sistema

3.7 Tras la instalación del sistema

¡Éxito! ¡Ha arrancado en el sistema Guix! De ahora en adelante, puede actualizar el sistema cuando quiera mediante la ejecución de, digamos:

```
  guix pull
  sudo guix system reconfigure /etc/config.scm
```

Esto construye una nueva generación del sistema con los últimos paquetes y servicios (véase Sección 8.14 [Invocación de guix system], página 399). Recomendamos realizarlo de manera regular de modo que su sistema incluya las últimas actualizaciones de seguridad (véase Capítulo 11 [Actualizaciones de seguridad], página 425).

Nota: Tenga en cuenta que `sudo guix` ejecuta el ejecutable `guix` de su usuaria y no el de root, ya que `sudo` no altera PATH. Para ejecutar explícitamente el ejecutable `guix` de root, escriba `sudo -i guix ...`

The difference matters here, because `guix pull` updates the `guix` command and package definitions only for the user it is ran as. This means that if you choose to use `guix system reconfigure` in root’s login shell, you’ll need to `guix pull` separately.

¡Unase a nosotras en #guix en la red IRC Freenode o en guix-devel@gnu.org para compartir su experiencia!

3.8 Instalación de Guix en una máquina virtual

Si desea instalar el sistema Guix en una máquina virtual (VM) o en un servidor privado virtual (VPS) en vez de en su preciada máquina, esta sección es para usted.

Si quiere arrancar una VM QEMU (https://qemu.org/) para instalar el sistema Guix en una imagen de disco, siga estos pasos:

1. Primero, obtenga y descomprima la imagen de instalación del sistema Guix como se ha descrito previamente (véase Sección 3.3 [Instalación desde memoria USB y DVD], página 24).

2. Cree una imagen de disco que contendrá el sistema instalado. Para crear una imagen de disco con formato qcow2, use la orden `qemu-img`:

   ```
   qemu-img create -f qcow2 guix-system.img 50G
   ```

   El fichero que obtenga será mucho menor de 50GB (típicamente menos de 1MB), pero crecerá cuando el dispositivo de almacenamiento virtualizado se vaya llenando.

3. Arranque la imagen de instalación USB en una máquina virtual:

   ```
   qemu-system-x86_64 -m 1024 -smp 1 -enable-kvm \
   -nic user,model=virtio-net-pci -boot menu=on,order=d \
   -drive file=guix-system.img \
   -drive media=cdrom,file=guix-system-install-1.0.1.sistema.iso
   ```

Está vacía inicialmente; otras contraseñas de usuarias tienen que ser inicializadas ejecutando la orden `passwd` como root, a menos que en su configuración se especifique de otra manera (véase [user-account-password], página 177). ¡Véase Sección 3.7 [Tras la instalación del sistema], página 33, para proceder a continuación!
-enable-kvm es opcional, pero mejora el rendimiento significativamente, véase Sección 8.16 [Ejecutar Guix en una máquina virtual], página 410.

4. Ahora es root en la VM, prosiga con el procedimiento de instalación. Véase Sección 3.4 [Preparación para la instalación], página 25, y siga las instrucciones.

Una vez complete la instalación, puede arrancar el sistema que está en la imagen guix-system.img. Véase Sección 8.16 [Ejecutar Guix en una máquina virtual], página 410, para información sobre cómo hacerlo.

### 3.9 Construcción de la imagen de instalación

La imagen de instalación descrita anteriormente se construyó usando la orden `guix system`, específicamente:

```
guix system disk-image --file-system-type=iso9660 \
gnu/system/install.scm
```

Eche un vistazo a `gnu/system/install.scm` en el árbol de fuentes, y vea también Sección 8.14 [Invocación de guix system], página 399, para más información acerca de la imagen de instalación.

### 3.10 Construcción de la imagen de instalación para placas ARM

Muchos dispositivos con procesador ARM necesitan una variante específica del cargador de arranque U-Boot (https://www.denx.de/wiki/U-Boot/).

Si construye una imagen de disco y el cargador de arranque no está disponible de otro modo (en otra unidad de arranque, etc.), es recomendable construir una imagen que incluya el cargador, específicamente:

```
guix system disk-image --system=armhf-linux -e '(@ (gnu system install) os-with-u-boot)
```

A20-OLinuXino-Lime2 es el nombre de la placa. Si especifica una placa no válida, una lista de placas posibles será mostrada.
4 Gestión de paquetes

El propósito de GNU Guix es permitir a las usuarias instalar, actualizar y borrar fácilmente paquetes de software, sin tener que conocer acerca de sus procedimientos de construcción o dependencias. Guix también va más allá de este conjunto obvio de características.

Este capítulo describe las principales características de Guix, así como las herramientas de gestión de paquetes que ofrece. Junto a la interfaz de línea de órdenes descrita a continuación (véase Sección 4.2 [Invocación de guix package], página 36, también puede usar la interfaz Emacs-Guix (véase The Emacs Guix Reference Manual), tras la instalación del paquete emacs-guix (ejecute la orden $M-x guix-help$ para iniciarse en su uso):

\begin{verbatim}
guix install emacs-guix
\end{verbatim}

4.1 Características

Cuando se usa Guix, cada paquete se encuentra en el *almacén de paquetes*, en su propio directorio—algo que se asemeja a `/gnu/store/xxx-paquete-1.2`, donde *xxx* es una cadena en base32.

En vez de referirse a estos directorios, las usuarias tienen su propio perfil, el cual apunta a los paquetes que realmente desean usar. Estos perfiles se almacenan en el directorio de cada usuaria, en `$HOME/.guix-profile`.

Por ejemplo, *alicia* instala GCC 4.7.2. Como resultado, `/home/alicia/.guix-profile/bin/gcc` apunta a `/gnu/store/...-gcc-4.7.2/bin/gcc`. Ahora, en la misma máquina, *rober* ha instalado ya GCC 4.8.0. El perfil de *rober* simplemente sigue apuntando a `/gnu/store/...-gcc-4.8.0/bin/gcc`—es decir, ambas versiones de GCC pueden coexistir en el mismo sistema sin ninguna interferencia.

La orden `guix package` es la herramienta central para gestión de paquetes (véase Sección 4.2 [Invocación de guix package], página 36). Opera en los perfiles de usuaria, y puede ser usada con privilegios de usuaria normal.

La orden proporciona las operaciones obvias de instalación, borrado y actualización. Cada invocación es en realidad una transacción: o bien la operación especificada se realiza satisfactoriamente, o bien nada sucede. Por tanto, si el proceso `guix package` es finalizado durante una transacción, o un fallo eléctrico ocurre durante la transacción, el perfil de usuaria permanece en su estado previo, y permanece usable.

Además, cualquier transacción de paquetes puede ser *vuelta atrás*. Si, por ejemplo, una actualización instala una nueva versión de un paquete que resulta tener un error importante, las usuarias pueden volver a la instancia previa de su perfil, de la cual se tiene constancia que funcionaba bien. De igual modo, la configuración global del sistema en Guix está sujeta a actualizaciones transaccionales y vuelta atrás (véase Sección 8.1 [Uso de la configuración del sistema], página 161).

Todos los paquetes en el almacén de paquetes pueden ser *eliminados por el recolector de basura*. Guix puede determinar a qué paquetes hacen referencia todavía los perfiles de usuarias, y eliminar aquellos que, de forma demostrable, no se haga referencia en ningún perfil (véase Sección 4.5 [Invocación de guix gc], página 48). Las usuarias pueden también borrar explícitamente generaciones antiguas de su perfil para que los paquetes a los que hacen referencia puedan ser recolectados.
Guix toma una aproximación puramente funcional en la gestión de paquetes, como se describe en la introducción (véase Capítulo 1 [Introducción], página 2). Cada nombre de directorio de paquete en /gnu/store contiene un hash de todas las entradas que fueron usadas para construir el paquete—compilador, bibliotecas, guiones de construcción, etc. Esta correspondencia directa permite a las usuarias asegurarse que una instalación dada de un paquete corresponde al estado actual de su distribución. Esto también ayuda a maximizar la reproducibilidad de la construcción: gracias al uso de entornos aislados de construcción, una construcción dada probablemente generará ficheros idénticos bit-a-bit cuando se realice en máquinas diferentes (véase Sección 2.5 [Invocación de guix-daemon], página 15).

Estos cimientos permiten a Guix ofrecer despliegues transparentes de binarios/fuentes. Cuando un binario pre-construido para un elemento de /gnu/store está disponible para descarga de una fuente externa—una sustitución, Guix simplemente lo descarga y desempaqueta; en otro caso construye el paquete de las fuentes, localmente (véase Sección 4.3 [Sustituciones], página 44). Debido a que los resultados de construcción son normalmente reproducibles bit-a-bit, las usuarias no tienen que confiar en los servidores que proporcionan sustituciones: pueden forzar una construcción local y retar a las proveedoras (véase Sección 7.11 [Invocación de guix challenge], página 153).

El control sobre el entorno de construcción es una característica que también es útil para desarrolladoras. La orden guix environment permite a desarrolladoras de un paquete configurar rápidamente el entorno de desarrollo correcto para su paquete, sin tener que instalar manualmente las dependencias del paquete en su perfil (véase Sección 5.1 [Invocación de guix environment], página 66).

Todo Guix y sus definiciones de paquetes están bajo control de versiones, y guix pull le permite “viajar en el tiempo” por la historia del mismo Guix (véase Sección 4.6 [Invocación de guix pull], página 51). Esto hace posible replicar una instancia de Guix en una máquina diferente o en un punto posterior del tiempo, lo que a su vez le permite replicar entornos de software completos, mientras que mantiene un preciso seguimiento de la procedencia del software.

### 4.2 Invocación de guix package

La orden guix package es la herramienta que permite a las usuarias instalar, actualizar y borrar paquetes, así como volver a configuraciones previas. Opera únicamente en el perfil propio de la usuaria, y funciona con privilegios de usuaria normal (véase Sección 4.1 [Características], página 35). Su sintaxis es:

```
guix package opciones
```

Primariamente, opciones especifica las operaciones a ser realizadas durante la transacción. Al completarse, un nuevo perfil es creado, pero las generaciones previas del perfil permanecen disponibles, en caso de que la usuaria quisiera volver atrás.

Por ejemplo, para borrar lua e instalar guile y guile-cairo en una única transacción:

```
guix package -r lua -i guile guile-cairo
```

Para su conveniencia, también se proporcionan los siguientes alias:

- guix search es un alias de guix package -s,
- guix install es un alias de guix package -i,
- guix remove es un alias de guix package -r,
• **guix upgrade** es un alias de **guix package -u**
• **y guix show** es un alias de **guix package --show**.

Estos alias tienen menos capacidad expresiva que **guix package** y proporcionan menos opciones, por lo que en algunos casos es probable que desee usar **guix package** directamente.

**guix package** también proporciona una **aproximación declarativa**, donde la usuaria especifica el conjunto exacto de paquetes a poner disponibles y la pasa a través de la opción **--manifest** (véase [profile-manifest], página 39).

Para cada usuaria, un enlace simbólico al perfil predeterminado de la usuaria es creado en $HOME/.guix-profile. Este enlace simbólico siempre apunta a la generación actual del perfil predeterminado de la usuaria. Por lo tanto, las usuarias pueden añadir $HOME/.guix-profile/bin a su variable de entorno PATH, y demás. Si no está usando el sistema Guix, considere la adición de las siguientes líneas en su ‘/.bash_profile’ (véase Sección “Bash Startup Files” en *The GNU Bash Reference Manual*) de manera que los nuevos shell que ejecute obtengan todas las definiciones correctas de las variables de entorno:

```
GUIX_PROFILE="$HOME/.guix-profile" ;
source "$HOME/.guix-profile/etc/profile"
```

En una configuración multiusuario, los perfiles de usuario se almacenan en un lugar registrado como una **raíz del sistema de ficheros**, a la que apunta $HOME/.guix-profile (véase Sección 4.5 [Invocación de guix gc], página 48). Ese directorio normalmente es localstatedir/guix/profiles/per-user/usuaria, donde localstatedir es el valor pasado a configure como **--localstatedir** y usuaria es el nombre de usuario. El directorio per-user se crea cuando se lanzan guix-daemon, y el subdirectorio usuaria es creado por guix package.

Las **opciones** pueden ser las siguientes:

```
--install=paquete ...
-i paquete ...
```

Instala los **paquetes** especificados.

Cada paquete puede especificar un nombre simple de paquete, como por ejemplo guile, o un nombre de paquete seguido por una arroba y el número de versión, como por ejemplo guile@1.8.8 o simplemente guile@1.8 (en el último caso la última versión con 1.8 como prefijo es seleccionada).

Si no se especifica un número de versión, la última versión disponible será seleccionada. Además, paquete puede contener dos puntos, seguido por el nombre de una de las salidas del paquete, como en gcc:doc o binutils@2.22:lib (véase Sección 4.4 [Paquetes con múltiples salidas], página 47). Los paquetes con el nombre correspondiente (y opcionalmente la versión) se buscan entre los módulos de la distribución GNU (véase Sección 6.1 [Módulos de paquetes], página 76).

A veces los paquetes tienen **entradas propagadas**: estas son las dependencias que se instalan automáticamente junto al paquete requerido (véase [package-propagated-inputs], página 82, para información sobre las entradas propagadas en las definiciones de paquete).

Un ejemplo es la biblioteca GNU MPC: sus ficheros de cabecera C hacen referencia a los de la biblioteca GNU MPFR, que a su vez hacen referencia a los de
la biblioteca GMP. Por tanto, cuando se instala MPC, las bibliotecas MPFR
y GMP también se instalan en el perfil; borrar MPC también borra MPFR y
GMP—a menos que también se hayan instalado explícitamente por la usuaria.

Por otra parte, los paquetes a veces dependen de la definición de variables de
entorno para sus rutas de búsqueda (véase a continuación la explicación de
\texttt{--search-paths}). Cualquier definición de variable de entorno que falte o sea
posiblemente incorrecta se informa aquí.

\begin{itemize}

\item \texttt{--install-from-expression=exp}\texttt{-e exp} \textit{Instala el paquete al que \texttt{exp} evalúa.}

\texttt{exp} debe ser una expresión Scheme que evalúe a un objeto \texttt{<package>}. Esta op\-ción es notablemente útil para diferenciar entre variantes con el mismo nombre
de paquete, con expresiones como \texttt{(\@ (gnu packages base) guile-final)}.

Fíjese que esta opción instala la primera salida del paquete especificado, lo cual
puede ser insuficiente cuando se necesita una salida específica de un paquete
con múltiples salidas.

\item \texttt{--install-from-file=fichero}\texttt{-f fichero} \textit{Instala el paquete que resulta de evaluar el código en \texttt{fichero}.}

Como un ejemplo, \texttt{fichero} puede contener una definición como esta (véase
Sección 6.2 \textit{Definición de paquetes}, página 77):

\begin{code}
(use-modules (guix)
  (guix build-system gnu)
  (guix licenses))

(package
  (name "hello")
  (version "2.10")
  (source (origin
    (method url-fetch)
    (uri (string-append "mirror://gnu/hello/hello-" version
           ".tar.gz"))
    (sha256
      (base32
        "0ssi1wpaf7plaswqjwigppsg5fyh99vdlb9kz17c91ng89ndq1i"))))
  (build-system gnu-build-system)
  (synopsis "Hello, GNU world: An example GNU package")
  (description "Guess what GNU Hello prints!")
  (home-page "http://www.gnu.org/software/hello/")
  (license gpl3+))
\end{code}

Las desarrolladoras pueden encontrarlo útil para incluir un fichero \texttt{guix.scm}
in la raíz del árbol de fuentes de su proyecto que puede ser usado para pro-
bar imágenes de desarrollo y crear entornos de desarrollo reproducibles (véase
Sección 5.1 \textit{Invocación de guix environment}, página 66).
--remove=paquete...
-r paquete...

Borra los paquetes especificados.
Como en --install, cada paquete puede especificar un número de versión y/o un nombre de salida además del nombre del paquete. Por ejemplo, -r glibc:debug eliminaría la salida debug de glibc.

--upgrade[=regexp ...]
-u [regexp ...]

Actualiza todos los paquetes instalados. Si se especifica una o más expresiones regular regexp, actualiza únicamente los paquetes instalados cuyo nombre es aceptado por regexp. Véase también la opción --do-not-upgrade más adelante. Tenga en cuenta que esto actualiza los paquetes a la última versión encontrada en la distribución instalada actualmente. Para actualizar su distribución, debe ejecutar regularmente guix pull (véase Sección 4.6 [Invocación de guix pull], página 51).

--do-not-upgrade[=regexp ...]

Cuando se usa junto a la opción --upgrade, no actualiza ningún paquete cuyo nombre sea aceptado por regexp. Por ejemplo, para actualizar todos los paquetes en el perfil actual excepto aquellos que contengan la cadena “emacs”:

$ guix package --upgrade . --do-not-upgrade emacs

--manifest=fichero
-m fichero

Crea una nueva generación del perfil desde el objeto de manifiesto devuelto por el código Scheme en fichero. Esta opción puede repetirse varias veces, en cuyo caso los manifiestos se concatenan.

Esto le permite declarar los contenidos del perfil en vez de construirlo a través de una secuencia de --install y órdenes similares. La ventaja es que fichero puede ponerse bajo control de versiones, copiarse a máquinas diferentes para reproducir el mismo perfil, y demás.

fichero debe devolver un objeto manifest, que es básicamente una lista de paquetes:

(terms->manifest
 (list guile-2.0
 ;; Usa una salida específica del paquete.
 (list guile-2.0 "debug")))

En este ejemplo tenemos que conocer qué módulos definen las variables emacs y guile-2.0 para proporcionar la línea use-package-modules correcta, lo cual puede ser complicado. En cambio podemos proporcionar especificaciones regulares de paquetes y dejar a specifications->manifest buscar los objetos de paquete correspondientes así:

(specifications->manifest
Capítulo 4: Gestión de paquetes

\texttt{\textbackslash('\texttt{\textbackslash"emacs\textbackslash"\textbackslash"guile@2.2\textbackslash"guile@2.2:debug\textquoteright\textbackslash\textbackslash\textbackslash})}

\textbf{--roll-back}

Vuelve a la generación previa del perfil—es decir, deshace la última transacción.
Cuando se combina con opciones como --install, la vuelta atrás ocurre antes que cualquier acción.
Cuando se vuelve atrás en la primera generación que realmente contiene paquetes instalados, se hace que el perfil apunte a la generación cero, la cual no contiene ningún fichero a excepción de sus propios metadatos.
Después de haber vuelto atrás, instalar, borrar o actualizar paquetes sobreescribe las generaciones futuras previas. Por tanto, la historia de las generaciones en un perfil es siempre linear.

\textbf{--switch-generation=patrón}

\texttt{-S patrón} Cambia a una generación particular definida por el patrón.

\textit{patrón} puede ser tanto un número de generación como un número prefijado con “+” o “-”. Esto último significa: mueve atrás/hacia delante el número especificado de generaciones. Por ejemplo, si quiere volver a la última generación antes de --roll-back, use --switch-generation=+1.

La diferencia entre --roll-back y --switch-generation=-1 es que --switch-generation no creará una generación cero, así que si la generación especificada no existe, la generación actual no se verá cambiada.

\textbf{--search-paths[=tipo]}

Informa de variables de entorno, en sintaxis Bash, que pueden necesitarse para usar el conjunto de paquetes instalado. Estas variables de entorno se usan para especificar las \textit{rutas de búsqueda} para ficheros usados por algunos de los paquetes.

Por ejemplo, GCC necesita que las variables de entorno \texttt{CPATH} y \texttt{LIBRARY_PATH} estén definidas para poder buscar cabeceras y bibliotecas en el perfil de la usauria (véase Sección “Environment Variables” en \textit{Using the GNU Compiler Collection (GCC)}). Si GCC y, digamos, la biblioteca de C están instaladas en el perfil, entonces --search-paths sugerirá fijar dichas variables a perfil/include y perfil/lib respectivamente.

El caso de uso tópico es para definir estas variables de entorno en el shell:

\texttt{$ eval \textbackslash\texttt{\textbackslash'guix package --search-paths\textbackslash'\textbackslash\textbackslash\textbackslash})}

\textit{tipo} puede ser exact, prefix o suffix, lo que significa que las definiciones de variables de entorno devueltas serán respectivamente las configuraciones exactas, prefijos o sufijos del valor actual de dichas variables. Cuando se omite, el valor predeterminado de \textit{tipo} es exact.

Esta opción puede usarse para calcular las rutas de búsqueda combinadas de varios perfiles. Considere este ejemplo:

\texttt{$ guix package -p foo -i guile}
\texttt{$ guix package -p bar -i guile-json}
\texttt{$ guix package -p foo -p bar --search-paths}

La última orden informa sobre la variable \texttt{GUILE_LOAD_PATH}, aunque, tomada individualmente, ni \texttt{foo ni bar} hubieran llevado a esa recomendación.
Capítulo 4: Gestión de paquetes

--profile=perfil
-p perfil Usa perfil en vez del perfil predeterminado de la usuaria. perfil debe ser el nombre de un fichero que se creará tras completar las tareas. Concretamente, perfil sera simplemente un enlace simbólico (“symlink”) que apunta al verdadero perfil en el que se instalan los paquetes:

$ guix install hello -p ~/código/mi-perfil
...

$ ~/código/mi-perfil/bin/hello
¡Hola mundo!

Todo lo necesario para deshacerse del perfil es borrar dicho enlace simbólico y sus enlaces relacionados que apuntan a generaciones específicas:

$ rm ~/código/mi-perfil ~/código/mi-perfil-*=link

--list-profiles
Enumera los perfiles de la usuaria:

$ guix package --list-profiles
/home/carlos/.guix-profile
/home/carlos/código/.guix-profile
/home/carlos/código/mi-perfil
/home/carlos/código/perfil-desarrollo
/home/carlos/tmp/prueba

Cuando se ejecuta como root, enumera todos los perfiles de todas las usuarias.

--allow-collisions
Permite colisiones de paquetes en el nuevo perfil. ¡Uselo bajo su propio riesgo!
Por defecto, guix package informa como un error las colisiones en el perfil. Las colisiones ocurren cuando dos o más versiones diferentes o variantes de un paquete dado se han seleccionado para el perfil.

--bootstrap
Use el Guile usado para el lanzamiento para construir el perfil. Esta opción es util únicamente a las desarrolladoras de la distribución.

Además de estas acciones, guix package acepta las siguientes opciones para consultar el estado actual de un perfil, o la disponibilidad de paquetes:

--search=regexp
-s regexp Enumera los paquetes disponibles cuyo nombre, sinopsis o descripción corresponde con regexp (sin tener en cuenta la capitalización), ordenados por relevancia. Imprime todos los metadatos de los paquetes coincidentes en formato recutils (véase GNU recutils manual).

Esto permite extraer campos específicos usando la orden recsel, por ejemplo:

$ guix package -s malloc | recsel -p name,version,relevance
name: jemalloc
version: 4.5.0
relevance: 6

name: glibc
version: 2.25
De manera similar, para mostrar el nombre de todos los paquetes disponibles bajo los términos de la GNU LGPL versión 3:

```
$ guix package -s "" | recsel -p name -e 'license ~ "LGPL 3"'
```

También es posible refinar los resultados de búsqueda mediante el uso de varias opciones -s, o varios parámetros a `guix search`. Por ejemplo, la siguiente orden devuelve una lista de juegos de mesa\(^1\) (esta vez mediante el uso del alias `guix search`:

```
$ guix search '\<board\>\' game | recsel -p name
```

Si omitimos -s `game`, también obtendríamos paquetes de software que tengan que ver con placas de circuitos impresos ("circuit board" en inglés); borrar los signos mayor y menor alrededor de `board` añadiría paquetes que tienen que ver con teclados (keyboard en inglés).

Y ahora para un ejemplo más elaborado. La siguiente orden busca bibliotecas criptográficas, descarta bibliotecas Haskell, Perl, Python y Ruby, e imprime el nombre y la sinopsis de los paquetes resultantes:

```
$ guix search crypto library | \
recsel -e '! (name ~ "^(ghc|perl|python|ruby)\")') -p name,synopsis
```

\(^1\) NdT: board en inglés.
name: python
version: 3.4.3

--list-installed[=regexp]
-I [regexp]
Enumera los paquetes actualmente instalados en el perfil especificado, con los últimos paquetes instalados mostrados al final. Cuando se especifica regexp, enumera únicamente los paquetes instalados cuyos nombres son aceptados por regexp.

Por cada paquete instalado, imprime los siguientes elementos, separados por tabuladores: el nombre del paquete, la cadena de versión, la parte del paquete que está instalada (por ejemplo, out para la salida predeterminada, include para sus cabeceras, etc.), y la ruta de este paquete en el almacén.

--list-available[=regexp]
-A [regexp]
List packages currently available in the distribution for this system (véase Sección 1.2 [Distribución GNU], página 3). When regexp is specified, list only available packages whose name matches regexp.

Por cada paquete, imprime los siguientes elementos separados por tabuladores: su nombre, su cadena de versión, las partes del paquete (véase Sección 4.4 [Paquetes con múltiples salidas], página 47) y la dirección de las fuentes de su definición.

--list-generations[=patrón]
-l [patrón]
Devuelve una lista de generaciones junto a sus fechas de creación; para cada generación, muestra los paquetes instalados, con los paquetes instalados más recientemente mostrados los últimos. Fíjese que la generación cero nunca se muestra.

Por cada paquete instalado, imprime los siguientes elementos, separados por tabuladores: el nombre de un paquete, su cadena de versión, la parte del paquete que está instalada (véase Sección 4.4 [Paquetes con múltiples salidas], página 47), y la ruta de este paquete en el almacén.

Cuando se usa patrón, la orden devuelve únicamente las generaciones que se ajustan al patrón. Entre los patrones adecuados se encuentran:

- **Enteros y enteros separados por comas.** Ambos patrones denotan números de generación. Por ejemplo, --list-generations=1 devuelve la primera. 
  Y --list-generations=1,8,2 devuelve las tres generaciones en el orden especificado. No se permiten ni espacios ni una coma al final.

- **Rangos.** --list-generations=2..9 imprime las generaciones especificadas y todas las intermedias. Fíjese que el inicio de un rango debe ser menor a su fin.

  También es posible omitir el destino final. Por ejemplo, --list-generations=2.. devuelve todas las generaciones empezando por la segunda.
- **Duraciones.** Puede también obtener los últimos $N$ días, semanas, o meses pasando un entero junto a la primera letra de la duración. Por ejemplo, `--list-generations=20d` enumera las generaciones que tienen hasta 20 días de antigüedad.

    `--delete-generations [=patrón]`

    Cuando se omite patrón, borra todas las generaciones excepto la actual.

    Esta orden acepta los mismos patrones que `--list-generations`. Cuando se especifica un patrón, borra las generaciones coincidentes. Cuando el patrón especifica una duración, las generaciones más antiguas que la duración especificada son las borradas. Por ejemplo, `--delete-generations=1m` borra las generaciones de más de un mes de antigüedad.

    Si la generación actual entra en el patrón, no es borrada. Tampoco la generación cero es borrada nunca.

    Fíjese que borrar generaciones previene volver atrás a ellas. Consecuentemente esta orden debe ser usada con cuidado.

Finalmente, ya que **guix package** puede lanzar procesos de construcción en realidad, acepta todas las opciones comunes de construcción (véase Sección 7.1.1 [Opciones comunes de construcción], página 119). También acepta opciones de transformación de paquetes, como `--with-source` (véase Sección 7.1.2 [Opciones de transformación de paquetes], página 122). No obstante, fíjese que las transformaciones del paquete se pierden al actualizar; para preservar las transformaciones entre actualizaciones, debe definir su propia variante del paquete en un módulo Guile y añadirlo a **GUIX_PACKAGE_PATH** (véase Sección 6.2 [Definición de paquetes], página 77).

### 4.3 Sustituciones

Guix permite despliegues transparentes de fuentes/binarios, lo que significa que puede tanto construir cosas localmente, como descargar elementos preconstruidos de un servidor, o ambas. Llamamos a esos elementos preconstruidos sustituciones—son sustituciones de los resultados de construcciones locales. En muchos casos, descargar una sustitución es mucho más rápido que construirla localmente.

Las sustituciones pueden ser cualquier cosa que resulte de una construcción de una derivación (véase Sección 6.5 [Derivaciones], página 101). Por supuesto, en el caso común, son paquetes binarios preconstruidos, pero los archivos de fuentes, por ejemplo, que también resultan de construcciones de derivaciones, pueden estar disponibles como sustituciones.

#### 4.3.1 Servidor oficial de sustituciones.

El servidor **ci.guix.gnu.org** es una fachada a una granja de construcción oficial que construye paquetes de Guix continuamente para algunas arquitecturas, y los pone disponibles como sustituciones. Esta es la fuente predeterminada de sustituciones; puede ser forzada a cambiar pasando la opción `--substitute-urls` bien a **guix-daemon** (véase [guix-daemon --substitute-urls], página 16) o bien a herramientas cliente como **guix package** (véase [client --substitute-urls option], página 120).

Las URLs de sustituciones pueden ser tanto HTTP como HTTPS. Se recomienda HTTPS porque las comunicaciones están cifradas; de modo contrario, usar HTTP hace visibles todas
las comunicaciones para alguien que las intercepte, quien puede usar la información obtenida para determinar, por ejemplo, si su sistema tiene vulnerabilidades de seguridad sin parchear.

El uso de sustituciones de la granja de construcción oficial se realiza de manera predeterminada cuando se usa el sistema Guix (véase Sección 1.2 [Distribución GNU], página 3). No obstante, no se realiza de manera predeterminada cuando se usa Guix en una distribución anfitriona, a menos que las active explícitamente vía uno de los pasos recomendados de instalación (véase Capítulo 2 [Instalación], página 5). Los siguientes párrafos describen cómo activar o desactivar las sustituciones para la granja oficial de construcción; el mismo procedimiento puede usarse para activar las sustituciones desde cualquier otro servidor que las proporcione.

### 4.3.2 Autorización de servidores de sustituciones

Para permitir a Guix descargar sustituciones de ci.guix.gnu.org o un espejo suyo, debe añadir su clave pública a la lista de control de acceso (ACL) de las importaciones de archivos, mediante el uso de la orden guix archive (véase Sección 4.11 [Invocación de guix archive], página 63). Hacerlo implica que confía que ci.guix.gnu.org no ha sido comprometido y proporciona sustituciones genuinas.

La clave pública para ci.guix.gnu.org se instala junto a Guix, en prefijo/share/guix/ci.guix.gnu.org.pub, donde prefijo es el prefijo de instalación de Guix. Si ha instalado Guix desde las fuentes, debe asegurarse de que comprobó la firma GPG de guix-1.0.1.tar.gz, el cual contiene el fichero de clave pública. Una vez hecho, puede ejecutar algo así:

```
# guix archive --authorize < prefijo/share/guix/ci.guix.gnu.org.pub
```

Una vez esté autorizada, la salida de una orden como guix build debería cambiar de algo como:

```
$ guix build emacs --dry-run
```

Se construirían las siguientes derivaciones:

```
/gnu/store/yr7bnx8wxcayd6j95r2clmkd11qh688w-emacs-24.3.drv
/gnu/store/x8qsh1hlhgx6csjwyobnv2i37z23w-dbus-1.6.4.tar.gz.drv
/gnu/store/1ixwp12fl950d15h2cj11c73733jay0z-alsa-lib-1.0.27.1.tar.gz.drv
```

... a algo así:

```
$ guix build emacs --dry-run
```

Se descargarían 112.3 MB:

```
/gnu/store/pk3n221bq6ydamyymqkkz7i69wiwjiwi-emacs-24.3
/gnu/store/2yn4ncnhpr61rssa6z0d9x22si0va3-libjpeg-8d
/gnu/store/7iyz6lg4azms9dn2mcjxaah9w77jq-cairo-1.12.16
```

... El texto ha cambiado de “Se construirían las siguientes derivaciones” a “Se descargarían 112.3 MB”. Esto indica que las sustituciones de ci.guix.gnu.org son usables y serán descargadas, cuando sea posible, en construcciones futuras.

El mecanismo de sustituciones puede ser desactivado globalmente ejecutando guix-daemon con --no-subsitutes (véase Sección 2.5 [Invocación de guix-daemon],
página 15). También puede ser desactivado temporalmente pasando la opción
--no-substitutes a guix package, guix build y otras herramientas de línea de órdenes.

4.3.3 Verificación de sustituciones

Guix detecta y emite errores cuando se intenta usar una sustitución que ha sido adulterada. Del mismo modo, ignora las sustituciones que no están firmadas, o que no están firmadas por una de las firmas enumeradas en la ACL.

No obstante hay una excepción: si un servidor no autorizado proporciona sustituciones que son idénticas bit-a-bit a aquellas proporcionadas por un servidor autorizado, entonces el servidor no autorizado puede ser usado para descargas. Por ejemplo, asumiendo que hemos seleccionado dos servidores de sustituciones con esta opción:

--substitute-url="https://a.example.org https://b.example.org"

Si la ACL contiene únicamente la clave para b.example.org, y si a.example.org resulta que proporciona exactamente las mismas sustituciones, Guix descargará sustituciones de a.example.org porque viene primero en la lista y puede ser considerado un espejo de b.example.org. En la práctica, máquinas de construcción independientes producen habitualmente los mismos binarios, gracias a las construcciones reproducibles bit-a-bit (véase a continuación).

Cuando se usa HTTPS, el certificado X.509 del servidor no se valida (en otras palabras, el servidor no está verificado), lo contrario del comportamiento habitual de los navegadores Web. Esto es debido a que Guix verifica la información misma de las sustituciones, como se ha explicado anteriormente, lo cual nos concierne (mientras que los certificados X.509 tratan de verificar las conexiones entre nombres de dominio y claves públicas).

4.3.4 Configuración de la pasarela.

Las sustituciones se descargan por HTTP o HTTPS. La variable de entorno http_proxy puede ser incluida en el entorno de guix-daemon y la respeta para las descargas de sustituciones. Fíjese que el valor de http_proxy en el entorno en que guix build, guix package y otras aplicaciones cliente se ejecuten no tiene ningún efecto.

4.3.5 Fallos en las sustituciones

Incluso cuando una sustitución de una derivación está disponible, a veces el intento de sustitución puede fallar. Esto puede suceder por varias razones: el servidor de sustituciones puede estar desconectado, la sustitución puede haber sido borrada, la conexión puede interrumpirse, etc.

Cuando las sustituciones están activadas y una sustitución para una derivación está disponible, pero el intento de sustitución falla, Guix intentará construir la derivación localmente dependiendo si se proporcionó la opción --fallback (véase [common build option --fallback], página 120). Específicamente, si no se pasó --fallback, no se realizarán construcciones locales, y la derivación se considera se considera fallida. No obstante, si se pasó --fallback, Guix intentará construir la derivación localmente, y el éxito o fracaso de la derivación depende del éxito o fracaso de la construcción local. Fíjese que cuando las sustituciones están desactivadas o no hay sustituciones disponibles para la derivación en cuestión, la construcción local se realizará siempre, independientemente de si se pasó la opción --fallback.
Para hacerse una idea de cuantas sustituciones hay disponibles en este momento, puede intentar ejecutar la orden `guix weather` (véase Sección 7.14 [Invocación de guix weather], página 157). Esta orden proporciona estadísticas de las sustituciones proporcionadas por un servidor.

### 4.3.6 Sobre la confianza en binarios

Hoy en día, el control individual sobre nuestra propia computación está a merced de instituciones, empresas y grupos con suficiente poder y determinación para subvertir la infraestructura de computación y explotar sus vulnerabilidades. Mientras que usar las sustituciones de `ci.guix.gnu.org` puede ser conveniente, recomendamos a las usuarias también construir sus paquetes, o incluso mantener su propia granja de construcción, de modo que `ci.guix.gnu.org` sea un objetivo menos interesante. Una manera de ayudar es publicando el software que construye usando `guix publish` de modo que otras tengan otro servidor más como opción para descargar sustituciones (véase Sección 7.10 [Invocación de guix publish], página 149).

Guix tiene los cimientos para maximizar la reproducibilidad de las construcciones (véase Sección 4.1 [Características], página 35). En la mayor parte de los casos, construcciones independientes de un paquete o derivación dada deben emitir resultados idénticos bit a bit. Por tanto, a través de un conjunto diverso de construcciones independientes de paquetes, podemos reforzar la integridad de nuestros sistemas. La orden `guix challenge` intenta ayudar a las usuarias en comprobar servidores de sustituciones, y asiste a las desarrolladoras encontrando construcciones no deterministas de paquetes (véase Sección 7.11 [Invocación de guix challenge], página 153). Similarmente, la opción `--check` de `guix build` permite a las usuarias si las sustituciones previamente instaladas son genuinas mediante su reconstrucción local (véase [build-check], página 126).

En el futuro, queremos que Guix permita la publicación y obtención de binarios hacia/desde otras usuarias, entre pares (P2P). En caso de interesarle hablar sobre este proyecto, unase a nosotras en `guix-devel@gnu.org`.

### 4.4 Paquetes con múltiples salidas

Habitualmente, los paquetes definidos en Guix tienen una salida única—es decir, el paquete de fuentes proporcionará exactamente un directorio en el almacén. Cuando se ejecuta `guix install glibc`, se instala la salida predeterminada del paquete GNU libc; la salida predeterminada se llama `out`, pero su nombre puede omitirse como se mostró en esta orden. En este caso particular, la salida predeterminada de `glibc` contiene todos ficheros de cabecera C, bibliotecas dinámicas, bibliotecas estáticas, documentación Info y otros ficheros auxiliares.

A veces es más apropiado separar varios tipos de ficheros producidos por un paquete único de fuentes en salidas separadas. Por ejemplo, la biblioteca C GLib (usada por GTK+ y paquetes relacionados) instala más de 20 MiB de documentación de referencia como páginas HTML. Para ahorrar espacio para usuarias que no la necesiten, la documentación va a una salida separada, llamada `doc`. Para instalar la salida principal de GLib, que contiene todo menos la documentación, se debe ejecutar:

```
guix install glib
```

La orden que instala su documentación es:
guix install glib:doc

Algunos paquetes instalan programas con diferentes “huellas de dependencias”. Por ejemplo, el paquete WordNet instala tanto herramientas de línea de órdenes como interfaces gráficas de usaria (IGU). Las primeras dependen únicamente de la biblioteca de C, mientras que las últimas dependen en Tcl/Tk y las bibliotecas de X subyacentes. En este caso, dejamos las herramientas de línea de órdenes en la salida predeterminada, mientras que las IGU están en una salida separada. Esto permite a las usuarias que no necesitan una IGU ahorrar espacio. La orden guix size puede ayudar a exponer estas situaciones (véase Sección 7.8 [Invocación de guix size], página 143). guix graph también puede ser útil (véase Sección 7.9 [Invocación de guix graph], página 145).

Hay varios de estos paquetes con salida múltiple en la distribución GNU. Otros nombres de salida convencionales incluyen lib para bibliotecas y posiblemente ficheros de cabecera, bin para programas independientes y debug para información de depuración (véase Capítulo 10 [Instalación de ficheros de depuración], página 424). La salida de los paquetes se enumera en la tercera columna del resultado de guix package --list-available (véase Sección 4.2 [Invocación de guix package], página 36).

4.5 Invocación de guix gc

Los paquetes instalados, pero no usados, pueden ser recolectados. La orden guix gc permite a las usuarias ejecutar explícitamente el recolector de basura para reclamar espacio del directorio /gnu/store—borrar ficheros o directorios manualmente puede dañar el almacén sin reparación posible!

El recolector de basura tiene un conjunto de raíces conocidas: cualquier fichero en /gnu/store alcanzable desde una raíz se considera vivo y no puede ser borrado; cualquier otro fichero se considera muerto y puede ser borrado. El conjunto de raíces del recolector de basura (“raíces del GC” para abreviar) incluye los perfiles predeterminados de las usuarias; por defecto los enlaces bajo /var/guix/gcroots representan dichas raíces. Por ejemplo, nuevas raíces del GC pueden añadirse con guix build --root (véase Sección 7.1 [Invocación de guix build], página 119). La orden guix gc --list-roots las enumera.

Antes de ejecutar guix gc --collect-garbage para liberar espacio, habitualmente es útil borrar generaciones antiguas de los perfiles de usaria; de ese modo, las construcciones antiguas de paquetes a las que dichas generaciones hacen referencia puedan ser reclamadas. Esto se consigue ejecutando guix package --delete-generations (véase Sección 4.2 [Invocación de guix package], página 36).

Nuestra recomendación es ejecutar una recolección de basura periódicamente, o cuando tenga poco espacio en el disco. Por ejemplo, para garantizar que al menos 5 GB están disponibles en su disco, simplemente ejecute:

    guix gc -F 5G

Es completamente seguro ejecutarla como un trabajo periódico no-interactivo (véase Sección 8.8.2 [Ejecución de tareas programadas], página 196, para la configuración de un trabajo de ese tipo). La ejecución de guix gc sin ningún parámetro recolectará tanta basura como se pueda, pero eso es no es normalmente conveniente: puede encontrarse teniendo que reconstruir o volviendo a bajar software que está “muerto” desde el punto de vista del recolector pero que es necesario para construir otras piezas de software—por ejemplo, la cadena de herramientas de compilación.
La orden `guix gc` tiene tres modos de operación: puede ser usada para recolectar ficheros muertos (predeterminado), para borrar ficheros específicos (la opción `--delete`), para mostrar información sobre la recolección de basura o para consultas más avanzadas. Las opciones de recolección de basura son las siguientes:

`--collect-garbage=[min]`

-`C [min]` Recolecta basura—es decir, ficheros no alcanzables de `/gnu/store` y subdirectorios. Esta operación es la predeterminada cuando no se especifican opciones.

  Cuando se proporciona `min`, para una vez que `min` bytes han sido recolectados. `min` puede ser un número de bytes, o puede incluir una unidad como sufijo, como `MiB` para mebibytes y `GB` para gigabytes (véase Sección “Block size” en GNU Coreutils).

  Cuando se omite `min`, recolecta toda la basura.

`--free-space=libre`

-`F libre` Recolecta basura hasta que haya espacio `libre` bajo `/gnu/store`, si es posible: `libre` denota espacio de almacenamiento, por ejemplo `500MiB`, como se ha descrito previamente.

  Cuando `libre` o más está ya disponible en `/gnu/store`, no hace nada y sale inmediatamente.

`--delete-generations[=duración]`

-`d [duración]` Antes de comenzar el proceso de recolección de basura, borra todas las generaciones anteriores a `duración`, para todos los perfiles de la usuaria; cuando se ejecuta como root esto aplica a los perfiles de `todas las usuarias`.

  Por ejemplo, esta orden borra todas las generaciones de todos sus perfiles que tengan más de 2 meses de antigüedad (excepto generaciones que sean las actuales), y una vez hecho procede a liberar espacio hasta que al menos 10 GiB estén disponibles:

  ```
  guix gc -d 2m -F 10G
  ```

`--delete`

-`D` Intenta borrar todos los ficheros del almacén y directorios especificados como parámetros. Esto falla si alguno de los ficheros no están en el almacén, o todavía están vivos.

`--list-failures` Enumera los elementos del almacén correspondientes a construcciones fallidas existentes en la caché.

  Esto no muestra nada a menos que el daemon se haya ejecutado pasando `--cache-failures` (véase Sección 2.5 [Invocación de guix-daemon], página 15).

`--list-roots` Enumera las raíces del recolector de basura poseídas por la usuaria; cuando se ejecuta como root, enumera `todas` las raíces del recolector de basura.
--list-busy
Enumerar los elementos del almacen que actualmente están siendo usados por procesos en ejecución. Estos elementos del almacen se consideran de manera efectiva raíces del recolector de basura: no pueden borrarse.

--clear-failures
Borra los elementos especificados del almacen de la caché de construcciones fallidas.
De nuevo, esta opción únicamente tiene sentido cuando el daemon se inicia con --cache-failures. De otro modo, no hace nada.

--list-dead
Muestra la lista de ficheros y directorios muertos todavía presentes en el almacen—es decir, ficheros y directorios que ya no se pueden alcanzar desde ninguna raíz.

--list-live
Muestra la lista de ficheros y directorios del almacen vivos.
Además, las referencias entre los ficheros del almacen pueden ser consultadas:

--references
--referrers
Enumerar las referencias (o, respectivamente, los referentes) de los ficheros del almacen pasados como parámetros.

--requisites
-R
Enumerar los requisitos los ficheros del almacen pasados como parámetros. Los requisitos incluyen los mismos ficheros del almacen, sus referencias, las referencias de estas, recursivamente. En otras palabras, la lista devuelta es la clausura transitiva de los ficheros del almacen.
Véase Sección 7.8 [Invocación de guix size], página 143, para una herramienta que perfila el tamaño de la clausura de un elemento. Véase Sección 7.9 [Invocación de guix graph], página 145, para una herramienta de visualización del grafo de referencias.

--derivers
Devuelve la/s derivación/es que conducen a los elementos del almacen dados (véase Sección 6.5 [Derivaciones], página 101).
Por ejemplo, esta orden:

guix gc --derivers 'guix package -I ^emacs$ | cut -f4'
devuelve el/los fichero/s .drv que conducen al paquete emacs instalado en su perfil.
Fíjese que puede haber cero ficheros .drv encontrados, por ejemplo porque estos ficheros han sido recolectados. Puede haber más de un fichero .drv encontrado debido a derivaciones de salida fija.

Por último, las siguientes opciones le permiten comprobar la integridad del almacen y controlar el uso del disco.

--verify[=opciones]
Verifica la integridad del almacen.
Por defecto, comprueba que todos los elementos del almacén marcados como válidos en la base de datos del daemon realmente existen en /gnu/store.

Cuando se proporcionan, opciones debe ser una lista separada por comas que contenga uno o más valores contents and repair.

Cuando se usa --verify=contents, el daemon calcula el hash del contenido de cada elemento del almacén y lo compara contra el hash de su base de datos. Las incongruencias se muestran como corrupciones de datos. Debido a que recorre todos los ficheros del almacén, esta orden puede tomar mucho tiempo, especialmente en sistemas con una unidad de disco lenta.

El uso de --verify=repair o --verify=contents,repair hace que el daemon intente reparar elementos corruptos del almacén obteniendo sustituciones para dichos elementos (véase Sección 4.3 [Sustituciones], página 44). Debido a que la reparación no es atómica, y por tanto potencialmente peligrosa, está disponible únicamente a la administradora del sistema. Una alternativa ligera, cuando sabe exactamente qué elementos del almacén están corruptos, es guix build --repair (véase Sección 7.1 [Invocación de guix build], página 119).

--optimize

Optimiza el almacén sustituyendo ficheros idénticos por enlaces duros—esto es la deduplicación.

El daemon realiza la deduplicación después de cada construcción satisfactoria o importación de archivos, a menos que se inicie con --disable-deduplication (véase Sección 2.5 [Invocación de guix-daemon], página 15). Por tanto, esta opción es útil primariamente cuando el daemon se estaba ejecutando con --disable-deduplication.

4.6 Invocación de guix pull

Los paquetes se instalan o actualizan con la última versión disponible en la distribución disponible actualmente en su máquina local. Para actualizar dicha distribución, junto a las herramientas de Guix, debe ejecutar guix pull: esta orden descarga el último código fuente de Guix y descripciones de paquetes, y lo despliega. El código fuente se descarga de un repositorio Git (https://git-scm.com), por defecto el repositorio oficial de GNU Guix, lo que no obstante puede ser personalizado.

Especificamente, guix pull descarga código de los canales (véase Sección 4.7 [Canales], página 54) especificados en una de las posibilidades siguientes, en este orden:

1. la opción --channels;
2. el fichero ~/.config/guix/channels.scm de la usuaria;
3. el fichero /etc/guix/channels.scm común al sistema;
4. los canales predeterminados en código especificados en la variable %default-channels.

Una vez completada, guix package usará paquetes y versiones de paquetes de esta copia recién obtenida de Guix. No solo eso, sino que todas las órdenes de Guix y los módulos Scheme también se tomarán de la última versión. Nuevas sub-órdenes guix incorporadas por la actualización también estarán disponibles.

Cualquier usuaria puede actualizar su copia de Guix usando guix pull, y el efecto está limitado a la usuaria que ejecute guix pull. Por ejemplo, cuando la usuaria root ejecuta
Capı́tulo 4: Gestión de paquetes

52

guix pull, dicha acción no produce ningún efecto en la versión del Guix que la usuaria
alicia ve, y viceversa.
El resultado de ejecutar guix pull es un perfil disponible bajo ~/.config/guix/current
conteniendo el último Guix. Por tanto, asegúrese de a~
nadirlo al inicio de sus rutas de
búsqueda de modo que use la última versión, de modo similar para el manual Info(véase
Capı́tulo 9 [Documentación], página 423).
export PATH="$HOME/.config/guix/current/bin:$PATH"
export INFOPATH="$HOME/.config/guix/current/share/info:$INFOPATH"
Las opciones --list-generations o -l enumeran las generaciones pasadas producidas
por guix pull, junto a detalles de su procedencia:
$ guix pull -l
Generación 1 10 jun 2018 00:18:18
guix 65956ad
URL del repositorio: https://git.savannah.gnu.org/git/guix.git
rama: origin/master
revisión: 65956ad3526ba09e1f7a40722c96c6ef7c0936fe
Generación 2 11 jun 2018 11:02:49
guix e0cc7f6
URL del repositorio: https://git.savannah.gnu.org/git/guix.git
rama: origin/master
revisión: e0cc7f669bec22c37481dd03a7941c7d11a64f1d
2 paquetes nuevos: keepalived, libnfnetlink
6 paquetes actualizados: emacs-nix-mode@2.0.4,
guile2.0-guix@0.14.0-12.77a1aac, guix@0.14.0-12.77a1aac,
heimdal@7.5.0, milkytracker@1.02.00, nix@2.0.4
Generación 3 13 jun 2018 23:31:07 (current)
guix 844cc1c
URL del repositorio: https://git.savannah.gnu.org/git/guix.git
rama: origin/master
revisión: 844cc1c8f394f03b404c5bb3aee086922373490c
28 paquetes nuevos: emacs-helm-ls-git, emacs-helm-mu, ...
69 paquetes actualizados: borg@1.1.6, cheese@3.28.0, ...
Sección 4.10 [Invocación de guix describe], página 62, para otras formas de describir el
estado actual de Guix.
El perfil ~/.config/guix/current funciona exactamente igual que los perfiles creados
por guix package (véase Sección 4.2 [Invocación de guix package], página 36). Es decir,
puede enumerar generaciones, volver a una generación previa—esto es, la versión anterior
de Guix—, etcétera:
$ guix pull --roll-back
se pasó de la generación 3 a la 2
$ guix pull --delete-generations=1
borrando /var/guix/profiles/per-user/carlos/current-guix-1-link


También puede usar `guix package` (véase Sección 4.2 [Invocación de guix package], página 36) para gestionar el perfil proporcionando su nombre de manera específica:

```
$ guix package -p ~/.config/guix/current --roll-back
se pasó de la generación 3 a la 2
$ guix package -p ~/.config/guix/current --delete-generations=1
borrando /var/guix/profiles/per-user/carlos/current-guix-1-link
```

La orden `guix pull` se invoca habitualmente sin parámetros, pero permite las siguientes opciones:

```
--url=url
--commit=revolución
--branch=rama

Descarga el código para el canal `guix` de la `url` especificada, en la `revolución` proporcionada (un ID de revisión Git representada como una cadena hexadecimal), o `rama`.

Estas opciones se proporcionan por conveniencia, pero también puede especificar su configuración en el fichero `~/.config/guix/channels.scm` o usando la opción `--channels` (vea más adelante).

```
--channels=fichero
-C fichero

Lee la lista de canales de `fichero` en vez de `~/.config/guix/channels.scm` o `/etc/guix/channels.scm`. `fichero` debe contener código Scheme que evalúe a una lista de objetos “channel”. Véase Sección 4.7 [Canales], página 54, para más información.

```
--news
-N

Muestra la lista de paquetes añadidos o actualizados desde la última generación, así como, ocasionalmente, noticias escritas por las autoras del canal para las usuarias (véase Sección 4.7 [Canales], página 54).

La información del paquete es la misma que la que se muestra cuando termina `guix pull`, pero sin la elipsis; es también similar a la salida de `guix pull -l` para la última generación (véase a continuación).

```
--list-generations[=patrón]
-1 [patrón]

Enumera todas las generaciones de `~/.config/guix/current` o, si se proporciona un `patrón`, el subconjunto de generaciones que correspondan con el `patrón`. La sintaxis de `patrón` es la misma que `guix package --list-generations` (véase Sección 4.2 [Invocación de guix package], página 36).

```
--roll-back

Vuelve a la generación previa de `~/.config/guix/current`—es decir, deshace la última transacción.

```
--switch-generation=patrón
-S patrón

Cambia a una generación particular definida por el `patrón`. `patrón` puede ser tanto un número de generación como un número prefijado con “+” o “-”. Esto último significa: mueve atrás/hacia delante el número especifici-
cado de generaciones. Por ejemplo, si quiere volver a la última generación antes de --roll-back, use --switch-generation=+1.

--delete-generations[=patrón]
-d [patrón]
Cuando se omite patrón, borra todas las generaciones excepto la actual.
Esta orden acepta los mismos patrones que --list-generations. Cuando se especifica un patrón, borra las generaciones coincidentes. Cuando el patrón especifica una duración, las generaciones más antiguas que la duración especificada son las borradas. Por ejemplo, --delete-generations=1m borra las generaciones de más de un mes de antigüedad.
Si la generación actual entra en el patrón, no será borrada.
Fíjese que borrar generaciones previene volver atrás a ellas. Consecuentemente esta orden debe ser usada con cuidado.
Sección 4.10 [Invocación de guix describe], página 62, para una forma de mostrar información sobre únicamente la generación actual.

--profile=perfil
-p perfil Usa perfil en vez de ~/.config/guix/current.

--dry-run
-n Muestra qué revisión/es del canal serían usadas y qué se construiría o sustituiría, sin efectuar ninguna acción real.

--system=sistema
-s sistema
Intenta construir paquetes para sistema—por ejemplo, x86_64-linux—en vez del tipo de sistema de la máquina de construcción.

--verbose
Produce salida prolija, escribiendo los logs de construcción por la salida de error estándar.

--bootstrap
Use el Guile usado para el lanzamiento para construir el último Guix. Esta opción es útil para las desarrolladoras de Guix únicamente.

El mecanismo de canales le permite instruir a guix pull de qué repositorio y rama obtener los datos, así como repositorios adicionales que contengan módulos de paquetes que deben ser desplegados. Véase Sección 4.7 [Canales], página 54, para más información.
Además, guix pull acepta todas las opciones de construcción comunes (véase Sección 7.1.1 [Opciones comunes de construcción], página 119).

4.7 Canales
Guix y su colección de paquetes son actualizados ejecutando guix pull (véase Sección 4.6 [Invocación de guix pull], página 51). Por defecto guix pull descarga y despliega el mismo Guix del repositorio oficial de GNU Guix. Esto puede ser personalizado definiendo canales en el fichero ~/.config/guix/channels.scm. Un canal especifica una URL y una rama de un repositorio Git para ser desplegado, y guix pull puede ser instruido para tomar los
Capítulo 4: Gestión de paquetes

4.7.1 Uso de un canal de Guix personalizado

El canal llamado guix especifica de donde el mismo Guix—sus herramientas de línea de órdenes y su colección de paquetes—debe ser descargado. Por ejemplo, suponga que quiere actualizar de su propia copia del repositorio Guix en example.org, y específicamente la rama super-hacks, para ello puede escribir en “/.config/guix/channels.scm esta especificación:

```scheme
(list (channel
   (name 'guix)
   (url "https://example.org/mi-guix.git")
   (branch "super-hacks"))
```

De aquí en adelante, guix pull obtendrá el código de la rama super-hacks del repositorio en example.org.

4.7.2 Especificación de canales adicionales

También puede especificar canales adicionales de los que obtener datos. Digamos que tiene un montón de variaciones personalizadas de paquetes que piensa que no tiene mucho sentido contribuir al proyecto Guix, pero quiere tener esos paquetes disponibles transparentemente en su línea de órdenes. Primero escribiría módulos que contengan esas definiciones de paquete (véase Sección 6.1 [Módulos de paquetes], página 76), los mantendría en un repositorio Git, y entonces usted y cualquier otra persona podrían usarlos como un canal adicional del que obtener paquetes. Limpio, ¿no?

Aviso: Antes de que, querida usuaria, grite—“¡Guau, esto es la caña!”—y publique su canal personal al mundo, nos gustaría compartir algunas palabras de precaución:

- Antes de publicar un canal, por favor considere contribuir sus definiciones de paquete al propio Guix (véase Capítulo 14 [Contribuir], página 435). Guix como proyecto es abierto a software libre de todo tipo, y los paquetes en el propio Guix están disponibles para todas las usuarias de Guix y se benefician del proceso de gestión de calidad del proyecto.

- Cuando mantiene definiciones de paquete fuera de Guix, nosotras, las desarrolladoras de Guix, consideramos que la carga de la compatibilidad cae de su lado. Recuerde que los módulos y definiciones de paquetes son solo código Scheme que usa varias interfaces programáticas (APIs). Queremos mantener la libertad de cambiar dichas interfaces para seguir mejorando Guix, posiblemente en formas que pueden romper su canal. Nunca cambiamos las interfaces gratuitamente, pero no vamos tampoco a congelar las interfaces.

- Corolario: si está usando un canal externo y el canal se rompe, por favor informe del problema a las autoras del canal, no al proyecto Guix.

¡Ha quedado advertida! Habiendo dicho esto, creemos que los canales externos son una forma práctica de ejercitar su libertad para aumentar la colección de paquetes de Guix y compartir su mejoras, que son pilares básicos del software.
libre (https://www.gnu.org/philosophy/free-sw.html). Por favor, envíenos un correo a guix-devel@gnu.org si quiere hablar sobre esto.

Para usar un canal, escriba en "~/config/guix/channels.scm" para instruir a guix pull para obtener datos de él además de los canales Guix predeterminados:

```scheme
;; Añade mis paquetes personales a aquellos que Guix provee.
(cons (channel
   (name 'mis-paquetes-personales)
   (url "https://example.org/paquetes-personales.git"))
%default-channels)
```

Fíjese que el fragmento previo es (¡como siempre!) código Scheme; usamos cons para añadir un canal a la lista de canales a la que la variable %default-channels hace referencia (véase Sección “Pairs” en GNU Guile Reference Manual). Con el fichero en este lugar, guix pull no sólo construye Guix sino también los módulos de paquetes de su propio repositorio. El resultado en "~/config/guix/current" es la unión de Guix con sus propios módulos de paquetes:

```
$ guix pull --list-generations
...
Generation 19 Aug 27 2018 16:20:48
  guix d894ab8
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: d894ab8e9bfabcefe6c49d9ba2e834dd5a73a300
  mis-paquetes-personales dd3df5e
    repository URL: https://example.org/paquetes-personales.git
    branch: master
    commit: dd3df5e2c8818760a8fc0bd699e55d3b69f0f2bb
11 new packages: mi-gimp, mi-emacs-con-cosas, ...
4 packages upgraded: emacs-racket-mode@0.0.2-2.1b78827, ...
```

La salida de guix pull previa muestra que la generación 19 incluye tanto Guix como paquetes del canal mis-paquetes-personales. Entre los paquetes nuevos y actualizados que son enumerados, algunos como mi-gimp y mi-emacs-con-cosas pueden venir de mis-paquetes-personales, mientras que otros vienen del canal predeterminado de Guix.

Para crear un canal, cree un repositorio Git que contenga sus propios módulos de paquetes y haga que esté disponible. El repositorio puede contener cualquier cosa, pero un canal útil contendrá módulos Guile que exportan paquetes. Una vez comience a usar un canal, Guix se comportará como si el directorio raíz del repositorio Git de dicho canal hubiese sido añadido a la ruta de carga de Guile (véase Sección “Load Paths” en GNU Guile Reference Manual). Por ejemplo, si su canal contiene un fichero en mis-paquetes/mis-herramientas.scm que define un módulo, entonces dicho módulo estará disponible bajo el nombre (mis-paquetes mis-herramientas), y podrá usarlo como cualquier otro módulo (véase Sección “Módulos” en GNU Guile Reference Manual).

### 4.7.3 Declaración de dependencias de canales

Las autoras de canales pueden decidir aumentar una colección de paquetes proporcionada por otros canales. Pueden declarar su canal como dependiente de otros canales en el fichero de metadatos .guix-channel, que debe encontrarse en la raíz del repositorio del canal.
Este fichero de metadatos debe contener una expresión-S simple como esta:

```
(channel
 (version 0)
 (dependencies
   (channel
     (name una-coleccion)
     (url "https://example.org/primera-coleccion.git"))
   (channel
     (name otra-coleccion)
     (url "https://example.org/segunda-coleccion.git")
     (branch "pruebas"))))
```

En el ejemplo previo, este canal se declara como dependiente de otros dos canales, que se obtendrán de manera automática. Los módulos proporcionados por el canal se compilarán en un entorno donde los módulos de todos estos canales declarados estén disponibles.

De cara a la confianza proporcionada y el esfuerzo que supondrá su mantenimiento, debería evitar depender de canales que no controle, y debería intentar minimizar el número de dependencias.

### 4.7.4 Módulos de paquetes en un subdirectorio

Como autora de un canal, es posible que desee mantener los módulos de su canal en un subdirectorio. Si sus módulos se encuentran en el subdirectorio `guix`, debe añadir un fichero `.guix-channel` de metadatos que contenga:

```
(channel
 (version 0)
 (directory "guix"))
```

### 4.7.5 Escritura de noticias del canal

Las autoras los canales pueden querer ocasionalmente comunicar información a sus usuarias acerca de cambios importantes en el canal. Podrían mandar un correo a todo el mundo, pero esto no es tan conveniente.

En vez de eso, los canales proporcionan un *fichero de noticias*; cuando las usuarias de un canal ejecutan `guix pull`, dicho fichero de noticias se lee automáticamente y `guix pull --news` puede mostrar los anuncios que correspondan a las nuevas revisiones que se han obtenido, si existen.

Para hacerlo, las autoras del canal deben declarar primero el nombre del fichero de noticias en su fichero `.guix-channel`:

```
(channel
 (version 0)
 (news-file "etc/noticias.txt"))
```

El fichero de noticias en sí, `etc/noticias.txt` en este ejemplo, debe ser similar a este:

```
(channel-news
 (version 0)
 (entry (tag "the-bug-fix")
   (title (en "Fixed terrible bug")
     (fr "Oh la la")
   ))
```
Este fichero consiste en una lista de entradas de noticias. Cada entrada se asocia a una revisión o una etiqueta: describe los cambios llevados a cabo en ella, y posiblemente también en revisiones anteriores. Las usuarias ven las entradas únicamente la primera vez que obtienen la revisión a la que la entrada hace referencia.

El campo del título (title) debe ser un resumen de una línea mientras que el cuerpo de la noticia (body) puede ser arbitrariamente largo, y ambos pueden contener marcas de Texinfo (véase Sección “Overview” en GNU Texinfo). Tanto el título como el cuerpo son una lista de tuplas de etiqueta de lengua y mensaje, lo que permite a guix pull mostrar las noticias en la lengua que corresponde a la localización de la usuaria.

Si desea traducir las noticias siguiendo un flujo de trabajo basado en gettext, puede extraer las cadenas traducibles con xgettext (véase Sección “xgettext Invocation” en GNU Gettext Utilities). Por ejemplo, asumiendo que escribe las entradas de noticias primero en inglés, la siguiente orden crea un fichero PO que contiene las cadenas a traducir:

```
xgettext -o news.po -l scheme -ken etc/news.scm
```

En resumen, sí, puede usar su canal como un blog. Pero tenga en cuenta que esto no sea exactamente lo que sus usuarias podrían esperar.

### 4.7.6 Replicación de Guix

La salida de guix pull --list-generations previa muestra precisamente qué revisiones se usaron para construir esta instancia de Guix. Por tanto podemos replicarla, digamos, en otra máquina, proporcionando una especificaciones de canales en ~/.config/guix/channels.scm que está “clavada” en estas revisiones:

```
;; Despliega unas revisiones específicas de mis canales de interés.
(list (channel
  (name 'guix)
  (url "https://git.savannah.gnu.org/git/guix.git")
  (commit "d894ab8e9bfabcefa6c49d9ba2e834dd5a73a300"))
  (channel
  (name 'mis-paquetes-personales)
  (url "https://example.org/paquetes-personales.git")
  (branch "dd3df5e2c8818760a8f0c0d699e55d3b69f6f2bb")))
```

La orden guix describe --format=channels puede incluso generar esta lista de canales directamente (véase Sección 4.10 [Invocación de guix describe], página 62). El fichero

\footnote{NdT: “entry” en inglés}
resultante puede usarse con la opción -C de guix pull (véase Sección 4.6 [Invocación de guix pull], página 51) o guix time-machine (véase Sección 4.8 [Invocación de guix time-machine], página 59).

En este punto las dos máquinas ejecutan exactamente el mismo Guix, con acceso a exactamente los mismos paquetes. La salida de guix build gimp en una máquina debe ser exactamente la misma, bit a bit, que la salida de la misma orden en la otra máquina. Esto también significa que ambas máquinas tienen acceso a todo el código fuente de Guix y, transitivamente, a todo el código fuente de cada paquete que define.

 Esto le proporciona superpoderes, lo que le permite seguir la pista de la procedencia de los artefactos binarios con un grano muy fino, y reproducir entornos de software a su voluntad—un tipo de capacidad de “meta-reproducibilidad”, si lo desea. Véase Sección 4.9 [Inferiores], página 60, para otro modo de tomar ventaja de estos superpoderes.

4.8 Invocación de guix time-machine

La orden guix time-machine proporciona acceso a otras revisiones de Guix, por ejemplo para instalar versiones antiguas de un paquete, o para reproducir una computación en un entorno idéntico. La revisión de Guix que se usará se define por el identificador de una revisión o por un fichero de descripción de canales creado con guix describe (véase Sección 4.10 [Invocación de guix describe], página 62).

La sintaxis general es:

```
guix time-machine opciones... -- orden param...
```

where command and arg... are passed unmodified to the guix command of the specified revision. The options that define this revision are the same as for guix pull (véase Sección 4.6 [Invocación de guix pull], página 51):

```
--url=url
--commit=revisión
--branch=rama
```

Usa el canal guix de la url especificada, en la revisión proporcionada (un ID de revisión Git representada como una cadena hexadecimal), o rama.

```
--channels=fichero
-C fichero
```

Lee la lista de canales de fichero. fichero debe contener código Scheme que evalúe a una lista de objetos “channel”. Véase Sección 4.7 [Canales], página 54, para más información.

Como con guix pull, la ausencia de opciones significa que se usará la última revisión de la rama master. Por tanto la orden

```
guix time-machine -- build hello
```

will thus build the package hello as defined in the master branch, which is in general a newer revision of Guix than you have installed. Time travel works in both directions!

Tenga en cuenta que guix time-machine puede desencadenar construcciones de canales y sus dependencias, y que pueden controlarse mediante las opciones de construcción estándar (véase Sección 7.1.1 [Opciones comunes de construcción], página 119).
4.9 Inferiores

Nota: La funcionalidad descrita aquí es una “versión de evaluación tecnológica” en la versión 1.0.1. Como tal, la interfaz está sujeta a cambios.

A veces necesita mezclar paquetes de revisiones de la revisión de Guix que está ejecutando actualmente con paquetes disponibles en una revisión diferente. Los inferiores de Guix le permiten conseguirlo componiendo diferentes revisiones de Guix de modo arbitrario.

Técnicamente, un “inferior” es esencialmente un proceso Guix separado conectado con su Guix principal a través de una sesión interactiva (véase Sección 6.8 [Invocación de guix repl], página 117). El módulo (guix inferior) le permite crear inferiores y comunicarse con ellos. También proporciona una interfaz de alto nivel para buscar y manipular los paquetes que un inferior proporciona—paquetes de inferiores.

Cuando se combina con los canales (véase Sección 4.7 [Canales], página 54), los inferiores proporcionan una forma simple de interactuar con una revisión separada de Guix. Por ejemplo, asumamos que desea instalar en su perfil el paquete guile actual, junto al paquete guile-json como existía en una revisión más antigua de Guix—quizá porque las versiones nuevas de guile-json tienen un API incompatible y quiere ejecutar su código contra la API antigua. Para hacerlo, puede escribir un manifiesto para usarlo con guix package --manifest (véase Sección 4.2 [Invocación de guix package], página 36); en dicho manifiesto puede crear un inferior para esa versión antigua de Guix que le interesa, y buscará el paquete guile-json en el inferior:

```
(use-modules (guix inferior) (guix channels)
             (srfi srfi-1)) ;para 'first'

(define channels
  ;; Esta es la revisión antigua de donde queremos extraer guile-json.
  (list (channel
          (name 'guix)
          (url "https://git.savannah.gnu.org/git/guix.git")
          (commit
           "65956ad3526ba09e1f7a40722c96c6ef7c0936fe"))))

(define inferior
  ;; Un inferior que representa la revisión previa.
  (inferior-for-channels channels)))

;; Ahora crea un manifiesto con el paquete "guile" actual
;; y el antiguo paquete "guile-json".
(packages->manifest
 (list (first (lookup-inferior-packages inferior "guile-json"))
       (specification->package "guile"))))
```

En su primera ejecución, guix package --manifest puede tener que construir el canal que especificó antes de crear el inferior; las siguientes ejecuciones serán mucho más rápidas porque la revisión de Guix estará en la caché.
El módulo (guix inferior) proporciona los siguientes procedimientos para abrir un inferior:

**inferior-for-channels canales [#:cache-directory] [#:ttl]**

Devuelve un inferior para `canales`, una lista de canales. Usa la caché en `cache-directory`, donde las entradas pueden ser reclamadas después de `ttl` segundos. Este procedimiento abre una nueva conexión al daemon de construcción.

Como efecto secundario, este procedimiento puede construir o sustituir binarios para `canales`, lo cual puede tomar cierto tiempo.

**open-inferior directorio [#:command "bin/guix"]**

Abre el Guix inferior en `directorio`, ejecutando `directorio/command repl` o su equivalente. Devuelve `#f` si el inferior no pudo ser ejecutado.

Los procedimientos enumerados a continuación le permiten obtener y manipular paquetes de inferiores.

**inferior-packages inferior**

Devuelve la lista de paquetes conocida por `inferior`.

**lookup-inferior-packages inferior nombre [versión]**

Devuelve la lista ordenada de paquetes del inferior que corresponden con `nombre` en `inferior`, con los números de versión más altos primero. Si `versión` tiene un valor verdadero, devuelve únicamente paquetes con un número de versión cuyo prefijo es `versión`.

**inferior-package? obj**

Devuelve verdadero si `obj` es un paquete inferior.

**inferior-package-name paquete**

**inferior-package-version paquete**

**inferior-package-synopsis paquete**

**inferior-package-description paquete**

**inferior-package-home-page paquete**

**inferior-package-location paquete**

**inferior-package-inputs paquete**

**inferior-package-native-inputs paquete**

**inferior-package-propagated-inputs paquete**

**inferior-package-transitive-propagated-inputs paquete**

**inferior-package-native-search-paths paquete**

**inferior-package-transitive-native-search-paths paquete**

**inferior-package-search-paths paquete**

Estos procedimientos son la contraparte de los accesos a los registros de paquete (véase Sección 6.2.1 [Referencia de package], página 81). La mayor parte funcionan interrogando al inferior del que `paquete` viene, por lo que el inferior debe estar vivo cuando llama a dichos procedimientos.
Los paquetes de inferiores pueden ser usados transparentemente como cualquier otro paquete u objeto-tipo-fichero en expresiones-G (véase Sección 6.7 [Expresiones-G], página 108). También se manejan transparentemente por el procedimiento \texttt{packages->manifest}, el cual se usa habitualmente en los manifiestos (véase Sección 4.2 [Invocación de guix package], página 36). Por tanto puede insertar un paquete de inferior prácticamente en cualquier lugar que pueda insertar un paquete normal: en manifiestos, en el campo \texttt{packages} de su declaración \texttt{operating-system}, etcétera.

4.10 Invocación de \texttt{guix describe}

A menudo desea responder a preguntas como: “¿Qué revisión de Guix estoy usando?” o “¿Qué canales estoy usando?” Esto es una información muy útil en muchas situaciones: si quiere \textit{replicar} un entorno en una máquina diferente o cuenta de usuaria, si desea informar de un error o determinar qué cambio en los canales que usa lo causó, o si quiere almacenar el estado de su sistema por razones de reproducibilidad. La orden \texttt{guix describe} responde a estas preguntas.

Cuando se ejecuta desde un \texttt{guix} bajado con \texttt{guix pull}, \texttt{guix describe} muestra el/los canal/es desde el/los que se construyó, incluyendo la URL de su repositorio y los IDs de las revisiones (véase Sección 4.7 [Canales], página 54):

```
$ guix describe
Generation 10 Sep 03 2018 17:32:44 (current)
guix e0fa68c
    repository URL: https://git.savannah.gnu.org/git/guix.git
    branch: master
    commit: e0fa68c7718fffd33d81af415279d6ddb518f727
```

Si está familiarizado con el sistema de control de versiones Git, esto es similar a \texttt{git describe}; la salida también es similar a la de \texttt{guix pull --list-generations}, pero limitada a la generación actual (véase Sección 4.6 [Invocación de guix pull], página 51). Debido a que el ID de revisión Git mostrado antes refiere sin ambigüedades al estado de Guix, esta información es todo lo necesario para describir la revisión de Guix que usa, y también para replicarla.

Para facilitar la replicación de Guix, también se le puede solicitar a \texttt{guix describe} devolver una lista de canales en vez de la descripción legible por humanos mostrada antes:

```
$ guix describe -f channels
(list (channel
    (name 'guix)
    (url "https://git.savannah.gnu.org/git/guix.git")
    (commit
      "e0fa68c7718fffd33d81af415279d6ddb518f727")))
```

Puede almacenar esto en un fichero y se lo puede proporcionar a \texttt{guix pull -C} en otra máquina o en un momento futuro, lo que instanciará \textit{esta revisión exacta de Guix} (véase Sección 4.6 [Invocación de guix pull], página 51). De aquí en adelante, ya que puede desplegar la misma revisión de Guix, puede también \textit{replicar un entorno completo de software}. Nosotras humildemente consideramos que esto es \textit{impresionante}, ¡y esperamos que le guste a usted también!
Los detalles de las opciones aceptadas por guix describe son las siguientes:

--format=\texttt{formato}  
\texttt{-f formato}  
Produce salida en el \texttt{formato} especificado, uno de:

\texttt{human} produce salida legible por humanos;
\texttt{channels} produce una lista de especificaciones de canales que puede ser pasada a guix pull -C o instalada como 
\texttt{~/.config/guix/channels.scm} (véase Sección 4.6 [Invocación de guix pull], página 51);
\texttt{json} produce una lista de especificaciones de canales en formato JSON;
\texttt{recutils} produce una lista de especificaciones de canales en formato Recutils.

--profile=\texttt{perfil}  
\texttt{-p perfil} Muestra información acerca del \texttt{perfil}.

4.11 Invocación de guix archive

La orden \texttt{guix archive} permite a las usuarias exportar ficheros del almacén en un único archivador, e importarlos posteriormente en una máquina que ejecute Guix. En particular, permite que los ficheros del almacén sean transferidos de una máquina al almacén de otra máquina.

Nota: Si está buscando una forma de producir archivos en un formato adecuado para herramientas distintas a Guix, véase Sección 5.2 [Invocación de guix pack], página 71.

Para exportar ficheros del almacén como un archivo por la salida estándar, ejecute:

\texttt{guix archive --export opciones especificaciones...}

\texttt{especificaciones} deben ser o bien nombres de ficheros del almacén o especificaciones de paquetes, como las de guix package (véase Sección 4.2 [Invocación de guix package], página 36). Por ejemplo, la siguiente orden crea un archivo que contiene la salida gui del paquete \texttt{git} y la salida principal de \texttt{emacs}:

\texttt{guix archive --export git:gui /gnu/store/...-emacs-24.3 > great.nar}

Si los paquetes especificados no están todavía construidos, \texttt{guix archive} los construye automáticamente. El proceso de construcción puede controlarse mediante las opciones de construcción comunes (véase Sección 7.1.1 [Opciones comunes de construcción], página 119).

Para transferir el paquete \texttt{emacs} a una máquina conectada por SSH, se ejecutaría:

\texttt{guix archive --export -r emacs | ssh otra-maquina guix archive --import}

De manera similar, un perfil de usauria completo puede transferirse de una máquina a otra de esta manera:

\texttt{guix archive --export -r $(readlink -f ~/.guix-profile) | \ ssh otra-maquina guix archive --import}

No obstante, fíjese que, en ambos ejemplos, todo \texttt{emacs} y el perfil como también todas sus dependencias son transferidas (debido a la \texttt{-r}), independiente de lo que estuviese ya disponible en el almacén de la máquina objetivo. La opción \texttt{--missing} puede ayudar a
esclarecer qué elementos faltan en el almacén objetivo. La orden guix copy simplifica y optimiza este proceso completo, así que probablemente es lo que debería usar en este caso (véase Sección 7.12 [Invocación de guix copy], página 155).

Los archivos se almacenan en el formato de “archivo normalizado” o “nar”, el cual es comparable a ‘tar’ en el espíritu, pero con diferencias que lo hacen más apropiado para nuestro propósito. Primero, en vez de almacenar todos los metadatos Unix de cada fichero, el formato nar solo menciona el tipo de fichero (normal, directorio o enlace simbólico); los permisos Unix y el par propietario/grupo se descartan. En segundo lugar, el orden en el cual las entradas de directorios se almacenan siempre siguen el orden de los nombres de ficheros de acuerdo a la ordenación de cadenas en la localización C. Esto hace la producción del archivo completamente determinista.

Durante la exportación, el daemon firma digitalmente los contenidos del archivo, y la firma digital se adjunta. Durante la importación, el daemon verifica la firma y rechaza la importación en caso de una firma inválida o si la clave firmante no está autorizada.

Las opciones principales son:

`--export` Exporta los ficheros del almacén o paquetes (véase más adelante). Escribe el archivo resultante a la salida estándar.

Las dependencias no están incluidas en la salida, a menos que se use `--recursive`.

`-r` `--recursive` Cuando se combina con `--export`, instruye a guix archive para incluir las dependencias de los elementos dados en el archivo. Por tanto, el archivo resultante está auto-contenido: contiene la clausura de los elementos exportados del almacén.

`--import` Lee un archivo de la entrada estándar, e importa los ficheros enumerados allí en el almacén. La operación se aborta si el archivo tiene una firma digital no válida, o si está firmado por una clave pública que no está entre las autorizadas (vea `--authorize` más adelante).

`--missing` Lee una lista de nombres de ficheros del almacén de la entrada estándar, uno por línea, y escribe en la salida estándar el subconjunto de estos ficheros que faltan en el almacén.

`--generate-key[=parámetros]` Genera un nuevo par de claves para el daemon. Esto es un prerequisito antes de que los archivos puedan ser exportados con `--export`. Tenga en cuenta que esta operación normalmente toma tiempo, ya que se necesita obtener suficiente entropía para generar un par de claves.

El par de claves generado se almacena típicamente bajo `/etc/guix`, en `signing-key.pub` (clave pública) y `signing-key.sec` (clave privada, que se debe mantener secreta). Cuando `parámetros` se omite, se genera una clave ECDSA usando la curva Ed25519, o, en versiones de Libgcrypt previas a la 1.6.0, es una clave RSA de 4096 bits. De manera alternativa, los `parámetros` pueden especificar parámetros `genkey` adecuados para Libgcrypt (véase
Sección “General public-key related Functions” en *The Libgcrypt Reference Manual*.

--authorize

Autoriza importaciones firmadas con la clave pública pasada por la entrada estándar. La clave pública debe estar en el “formato avanzado de expresiones-s”—es decir, el mismo formato que el fichero `signing-key.pub`.

La lista de claves autorizadas se mantiene en el fichero editable por personas `/etc/guix/acl`. El fichero contiene “expresiones-s en formato avanzado” ([https://people.csail.mit.edu/rivest/Sexp.text](https://people.csail.mit.edu/rivest/Sexp.text)) y está estructurado como una lista de control de acceso en el formato Infraestructura Simple de Clave Pública (SPKI) ([https://theworld.com/~cme/spki.txt](https://theworld.com/~cme/spki.txt)).

--extract=directorio

-x directorio

Lee un único elemento del archivo como es ofrecido por los servidores de sustituciones (véase Sección 4.3 [Sustituciones], página 44) y lo extrae a `directorio`. Esta es una operación de bajo nivel necesitada únicamente para casos muy concretos; véase a continuación.

Por ejemplo, la siguiente orden extrae la sustitución de Emacs ofrecida por ci.guix.gnu.org en `/tmp/emacs`:

```
$ wget -O - \ 
  https://ci.guix.gnu.org/nar/...-emacs-24.5 \ 
  | bunzip2 | guix archive -x /tmp/emacs
```

Los archivos de un único elemento son diferentes de los archivos de múltiples elementos producidos por `guix archive --export`; contienen un único elemento del almacén, y no embeben una firma. Por tanto esta operación no verifica la firma y su salida debe considerarse insegura.

El propósito primario de esta operación es facilitar la inspección de los contenidos de un archivo que provenga probablemente de servidores de sustituciones en los que no se confía.

--list

-t

Lee un único elemento del archivo como es ofrecido por los servidores de sustituciones (véase Sección 4.3 [Sustituciones], página 44) e imprime la lista de ficheros que contiene, como en este ejemplo:

```
$ wget -O - \ 
  https://ci.guix.gnu.org/nar/lzip/...-emacs-26.3 \ 
  | lzip -d | guix archive -t
```
5 Desarrollo

Si es una desarrolladora de software, Guix le proporciona herramientas que debería encontrar útiles—independientemente del lenguaje en el que desarrolle actualmente. Esto es sobre lo que trata este capítulo.

La orden `guix environment` proporciona una manera conveniente de configurar un entorno de desarrollo que contenga todas las dependencias y herramientas necesarias para trabajar en el paquete de software de su elección. La orden `guix pack` le permite crear aplicaciones empaquetadas que pueden ser distribuidas con facilidad a usuarias que no usen Guix.

5.1 Invocación de `guix environment`

El propósito de `guix environment` es ayudar a las hackers en la creación de entornos de desarrollo reproducibles sin modificar los paquetes de su perfil. La herramienta `guix environment` toma uno o más paquetes, construye todas sus entradas y crea un entorno shell para usarlos.

La sintaxis general es:

```
guix environment opciones paquete...
```

El ejemplo siguiente lanza un nuevo shell preparado para el desarrollo de GNU Guile:

```
guix environment guile
```

Si las dependencias necesarias no están construidas todavía, `guix environment` las construye automáticamente. El entorno del nuevo shell es una versión aumentada del entorno en el que `guix environment` se ejecutó. Contiene las rutas de búsqueda necesarias para la construcción del paquete proporcionado añadidas a las variables ya existentes. Para crear un entorno “puro”, donde las variables de entorno previas no existen, use la opción `--pure`.

`guix environment` define la variable `GUIX_ENVIRONMENT` en el shell que lanza; su valor es el nombre de fichero del perfil para este entorno. Esto permite a las usuarias, digamos, definir un prompt para entornos de desarrollo en su `.bashrc` (véase Sección “Bash Startup Files” en The GNU Bash Reference Manual):

```
if [ -n "$GUIX_ENVIRONMENT" ]
then
    export PS1="\u@\h \w [dev]\$ 
fi
```

... o para explorar el perfil:

```
$ ls "$GUIX_ENVIRONMENT/bin"
```

Adicionalmente, más de un paquete puede ser especificado, en cuyo caso se usa la unión de las entradas de los paquetes proporcionados. Por ejemplo, la siguiente orden lanza un shell donde todas las dependencias tanto de Guile como de Emacs están disponibles:

```
guix environment guile emacs
```

---

1 Las usuarias habitualmente aumentan de forma incorrecta las variables de entorno como `PATH` en su fichero `~/.bashrc`. Como consecuencia, cuando `guix environment` se ejecuta, Bash puede leer `~/.bashrc`, por tanto introduciendo “impurezas” en esas variables de entorno. Es un error definir dichas variables de entorno en `~/.bashrc`; en vez de ello deben definirse en `~/.bash_profile`, el cual es únicamente cargado por el shell de ingreso al sistema. Véase Sección “Bash Startup Files” en The GNU Bash Reference Manual, para detalles sobre los ficheros de inicio de Bash.
A veces no se desea una sesión interactiva de shell. Una orden arbitraria se puede invocar usando el valor `--` para separar la orden del resto de los parámetros:

```
guix environment guile -- make -j4
```

En otras situaciones, es más conveniente especificar una lista de paquetes necesarios en el entorno. Por ejemplo, la siguiente orden ejecuta `python` desde un entorno que contiene Python 2.7 y NumPy:

```
guix environment --ad-hoc python2-numpy python-2.7 -- python
```

Más allá, se pueden desear las dependencias de un paquete y también algunos paquetes adicionales que no son dependencias ni en tiempo de construcción ni en el de ejecución, pero son útiles no obstante para el desarrollo. Por esta razón, la opción `--ad-hoc` es posicional. Los paquetes que aparecen antes de `--ad-hoc` se interpretan como paquetes cuyas dependencias se añadirán al entorno. Los paquetes que aparecen después se interpretan como paquetes que se añadirán directamente al entorno. Por ejemplo, la siguiente orden crea un entorno de desarrollo Guix que incluye adicionalmente Git y strace:

```
guix environment --pure guix --ad-hoc git strace
```

En ocasiones es deseable aislar el entorno tanto como sea posible, para obtener la máxima pureza y reproducibilidad. En particular, cuando se usa Guix en una distribución anfitriona que no es el sistema Guix, es deseable prevenir acceso a `/usr/bin` y otros recursos del sistema desde el entorno de desarrollo. Por ejemplo, la siguiente orden lanza un REPL Guile en un “contenedor” donde únicamente el almacen y el directorio actual están montados:

```
guix environment --ad-hoc --container guile -- guile
```

Nota: La opción `--container` requiere Linux-libre 3.19 o más nuevo.

Las opciones disponibles se resumen a continuación.

```
--root=fichero
-r fichero
```

Hace que `fichero` sea un enlace simbólico al perfil para este entorno, y lo registra como una raíz del recolector de basura.

Esto es útil si desea proteger su entorno de la recolección de basura, hacerlo “persistente”.

Cuando se omite esta opción, el entorno se protege de la recolección de basura únicamente por la duración de la sesión `guix environment`. Esto significa que la siguiente vez que vuelva a crear el mismo entorno, puede tener que reconstruir o volver a descargar paquetes. Véase Sección 4.5 [Invocación de guix gc], página 48, para más información sobre las raíces del recolector de basura.

```
--expression=expr
-e expr
```

Crea un entorno para el paquete o lista de paquetes a los que evalúa `expr`. Por ejemplo, ejecutando:

```
guix environment -e '(@ (gnu packages maths) petsc-openmpi)'
```

inicia un shell con el entorno para esta variante específica del paquete PETSc.

Ejecutar:

```
guix environment --ad-hoc -e '(@ (gnu) %base-packages)'
```

inicia un shell con todos los paquetes básicos del sistema disponibles.
Las órdenes previas usan únicamente la salida predeterminada de los paquetes dados. Para seleccionar otras salidas, tuplas de dos elementos pueden ser especificadas:

```
guix environment --ad-hoc -e '(list (@ (gnu packages bash) bash) "include")'
```

```
--load=fichero
-l fichero
```

Crea un entorno para el paquete o la lista de paquetes a la que el código en `fichero` evalúa.

Como un ejemplo, `fichero` puede contener una definición como esta (véase Sección 6.2 [Definición de paquetes], página 77):

```
(use-modules (guix)
 (gnu packages gdb)
 (gnu packages autotools)
 (gnu packages texinfo))
```

```
;; Augment the package definition of GDB with the build tools
;; needed when developing GDB (and which are not needed when
;; simply installing it.)
(package (inherit gdb)
 (native-inputs '(("autoconf" ,autoconf-2.64)
 ("automake" ,automake)
 ("texinfo" ,texinfo)
 ,@(package-native-inputs gdb))))
```

```
--manifest=fichero
-m fichero
```

Crea un entorno para los paquetes contenidos en el objeto manifest devuelto por el código Scheme en `file`. Esta opción se puede repetir varias veces, en cuyo caso los manifiestos se concatenan.

 Esto es similar a la opción del mismo nombre en `guix package` (véase [profile-manifest], página 39) y usa los mismos ficheros de manifiesto.

```
--ad-hoc
```

Incluye todos los paquetes especificados en el entorno resultante, como si un paquete ad hoc hubiese sido definido con ellos como entradas. Esta opción es útil para la creación rápida un entorno sin tener que escribir una expresión de paquete que contenga las entradas deseadas.

Por ejemplo, la orden:

```
guix environment --ad-hoc guile guile-sdl -- guile
```

ejecuta `guile` en un entorno donde están disponibles Guile y Guile-SDL.

Fíjese que este ejemplo solicita implícitamente la salida predeterminada de `guile` y `guile-sdl`, pero es posible solicitar una salida específica—por ejemplo, `glib:bin` solicita la salida `bin` de `glib` (véase Sección 4.4 [Paquetes con múltiples salidas], página 47).

Esta opción puede componerse con el comportamiento predeterminado de `guix environment`. Los paquetes que aparecen antes de `--ad-hoc` se interpretan como paquetes cuyas dependencias se añadirán al entorno, el comportamiento
predefinido. Los paquetes que aparecen después se interpretan como paquetes a añadir directamente al entorno.

--pure

Olvida las variables de entorno existentes cuando se construye un nuevo entorno, excepto aquellas especificadas con --preserve (véase más adelante). Esto tiene el efecto de crear un entorno en el que las rutas de búsqueda únicamente contienen las entradas del paquete.

--preserve=regexp

Cuando se usa junto a --pure, preserva las variables de entorno que corresponden con regexp—en otras palabras, las pone en una lista de variables de entorno que deben preservarse. Esta opción puede repetirse varias veces.

```
guix environment --pure --preserve=^SLURM --ad-hoc openmpi ...
 -- mpirun ...
```

Este ejemplo ejecuta mpirun en un contexto donde las únicas variables de entorno definidas son PATH, variables de entorno cuyo nombre empiece con SLURM, así como las variables “preciosas” habituales (HOME, USER, etc.).

--search-paths

Muestra las definiciones de variables de entorno que componen el entorno.

--system=sistema

-s sistema

Intenta construir para sistema—por ejemplo, i686-linux.

--container

-C

Ejecuta la orden en un contenedor aislado. El directorio actual fuera del contenedor es asociado al interior del contenedor. Adicionalmente, a menos que se fuerce con --user, un directorio de prueba de la usuaria se crea de forma que coincida con el directorio actual de la usuaria, y /etc/passwd se configura adecuadamente.

El proceso lanzado se ejecuta como el usuario actual fuera del contenedor. Dentro del contenedor, tiene el mismo UID y GID que el usuario actual, a menos que se proporcione --user (véase más adelante).

--network

-N

Para contenedores, comparte el espacio de nombres de red con el sistema anfitrión. Los contenedores creados sin esta opción únicamente tienen acceso a la red local.

--link-profile

-P

Para contenedores, enlaza el perfil del entorno a ~/.guix-profile dentro del contenedor. Es equivalente a la ejecución de ln -s $GUIX_ENVIRONMENT ~/.guix-profile dentro del contenedor. El enlace fallará e interrumpirá el entorno si el directorio ya existe, lo cual será probablemente el caso si guix environment se invocó en el directorio de la usuaria.
Determinados paquetes se configuran para buscar en `~/.guix-profile` ficheros de configuración y datos; `--link-profile` permite a estos programas operar de la manera esperada dentro del entorno.

```
--user=usuaria
-u usuaria
```

Para contenedores, usa el nombre de usuaria `usuaria` en vez de la actual. La entrada generada en `/etc/passwd` dentro del contenedor contendrá el nombre `usuaria`; su directorio será `/home/usuaria` y ningún dato GECOS de la usuaria se copiará. Más aún, el UID y GID dentro del contenedor son 1000. `usuaria` no debe existir en el sistema.

Adicionalmente, cualquier ruta compartida o expuesta (véanse `--share` y `--expose` respectivamente) cuyo destino esté dentro de la carpeta actual de la usuaria será reasociada en relación a `/home/usuaria`; esto incluye la relación automática del directorio de trabajo actual.

```
# expondrá las rutas /home/foo/ddt, /home/foo/prueba y /home/foo/objetivo
cd $HOME/ddt
guix environment --container --user=foo \
  --expose=$HOME/prueba \
  --expose=/tmp/objetivo=$HOME/objetivo
```

Mientras esto limita el escape de la identidad de la usuaria a través de las rutas de sus directorios y cada uno de los campos de usuaria, esto es únicamente un componente útil de una solución de privacidad/anonimato más amplia—no una solución completa.

```
--no-cwd
```

El comportamiento predeterminado con contenedores es compartir el directorio de trabajo actual con el contenedor aislado e inmediatamente cambiar a dicho directorio dentro. Si esto no es lo deseado, `--no-cwd` implica que el directorio actual `no` se compartirá automáticamente y, en vez de cambiar a dicho directorio, se cambiará al directorio de la usuaria dentro del contenedor. Véase también `--user`.

```
--expose=fuente[=destino]
--share=fuente[=destino]
```

For containers, `--expose` (resp. `--share`) exposes the file system `source` from the host system as the read-only (resp. writable) file system `target` within the container. If `target` is not specified, `source` is used as the target mount point in the container.

El ejemplo a continuación lanza una sesión interactiva de Guile en un contenedor donde el directorio principal de la usuaria es accesible en modo solo-lectura a través del directorio `/intercambio`:

```
guix environment --container --expose=$HOME=/intercambio --ad-hoc guile --
guide...
```

Además, `guix environment` acepta todas las opciones comunes de construcción que permite `guix build` (véase Sección 7.1.1 [Opciones comunes de construcción], página 119) así

---

2 Por ejemplo, el paquete `fontconfig` inspecciona `~/.guix-profile/share/fonts` en busca de nuevas tipografías.
como las opciones de transformación de paquetes (véase Sección 7.1.2 [Opciones de transformación de paquetes], página 122).

5.2 Invocación de guix pack

De manera ocasionales querrá dar software a gente que (¡todavía!) no tiene la suerte de usar Guix. Usted les diría que ejecuten `guix package -i algo`, pero eso no es posible en este caso. Aquí es donde viene `guix pack`.

Nota: Si está buscando formas de intercambiar binarios entre máquinas que ya ejecutan Guix, véase Sección 7.12 [Invocación de guix copy], página 155, Sección 7.10 [Invocación de guix publish], página 149, y Sección 4.11 [Invocación de guix archive], página 63.

La orden `guix pack` crea un paquete reducido o empaquetado de software: crea un archivador tar u otro tipo que contiene los binarios del software en el que está interesada y todas sus dependencias. El archivo resultante puede ser usado en una máquina que no tiene Guix, y la gente puede ejecutar exactamente los mismos binarios que usted tiene con Guix. El paquete en sí es creado de forma reproducible bit-a-bit, para que cualquiera pueda verificar que realmente contiene los resultados de construcción que pretende distribuir.

Por ejemplo, para crear un empaquetado que contenga Guile, Emacs, Geiser y todas sus dependencias, puede ejecutar:

```
$ guix pack guile emacs geiser
... 
/gnu/store/...-pack.tar.gz
```

El resultado aquí es un archivador tar que contiene un directorio de `/gnu/store` con todos los paquetes relevantes. El archivador resultante contiene un perfil con los tres paquetes de interés; el perfil es el mismo que se hubiera creado por `guix package -i`. Este es el mecanismo usado para crear el propio archivador de binarios separado de Guix (véase Sección 2.1 [Instalación binaria], página 5).

Las usuarias de este empaquetado tendrán que ejecutar `/gnu/store/...-profile/bin/guile` para ejecutar guile, lo que puede resultar inconveniente. Para evitarlo, puede crear, digamos, un enlace simbólico `/opt/gnu/bin` al perfil:

```
guix pack -S /opt/gnu/bin=bin guile emacs geiser
```

De este modo, las usuarias pueden escribir alegremente `/opt/gnu/bin/guile` y disfrutar.

¿Qué pasa si la receptora de su paquete no tiene privilegios de root en su máquina y por lo tanto no puede desempaquetarlo en la raíz del sistema de ficheros? En ese caso, lo que usted desea es usar la opción `--relocatable` (véase a continuación). Esta opción produce binarios reposicionables, significando que pueden ser colocados en cualquier lugar de la jerarquía del sistema de ficheros: en el ejemplo anterior, las usuarias pueden desempaquetar el archivador en su directorio de usuario y ejecutar directamente `/opt/gnu/bin/guile`.

De manera alternativa, puede producir un empaquetado en el formato de imagen Docker usando la siguiente orden:

```
guix pack -f docker -S /bin=bin guile guile-readline
```

El resultado es un archivador “tar” que puede ser proporcionado a la orden `docker load`, seguida de `docker run`:

```
docker load < fichero
```
docker run -ti guile-guile-readline /bin/guile


Otra opción más es producir una imagen SquashFS con la siguiente orden:

    guix pack -f squashfs bash guile emacs geiser

El resultado es una imagen de sistema de ficheros SquashFS que puede ser o bien montada, o bien usada directamente como una imagen contenedora de sistemas de ficheros con el entorno de ejecución de contenedores Singularity (https://singularity.lbl.gov), usando órdenes como singularity shell o singularity exec.

Varias opciones de la línea de órdenes le permiten personalizar su empaquetado:

--format=formato
-f formato

Produce un empaquetado en el formato específico.

Los formatos disponibles son:

tarball Es el formato predeterminado. Produce un archiver que contiene todos los binarios y enlaces simbólicos especificados.

docker Produce un archiver que sigue la especificación de imágenes Docker (https://github.com/docker/docker/blob/master/image/spec/v1.2.md). El “nombre de repositorio” como aparece en la salida de la orden docker images se calcula a partir de los nombres de paquete proporcionados en la línea de órdenes o en el fichero de manifiesto.

squashfs Produce una imagen SquashFS que contiene todos los binarios y enlaces simbólicos especificados, así como puntos de montaje vacíos para sistemas de ficheros virtuales como procfs.

Nota: Singularity necesita que proporcione /bin/sh en la imagen. Por esta razón, guix pack -f squashfs siempre implica -S /bin=bin. Por tanto, su invocación de guix pack debe siempre comenzar de manera similar a esta:

    guix pack -f squashfs bash ...

Si se olvida del paquete bash (o similar), singularity run y singularity exec fallarán con el mensaje “no existe el fichero o directorio”, lo que no sirve de ayuda.

--relocatable
-R

Produce binarios reposicionables—es decir, binarios que se pueden encontrar en cualquier lugar de la jerarquía del sistema de ficheros, y ejecutarse desde allí.

Cuando se proporciona una vez la opción, los binarios resultantes necesitan la implementación de espacios de nombres de usuaria del núcleo Linux; cuando se proporciona dos veces\(^3\), los binarios reposicionables usan PRoot si los espa-

\(^3\) Esto es un truco para memorizarlo: -RR, que añade PRoot, puede pensarse como “Realmente Reposicionable”. Curioso, ¿no?
cios de nombres de usuaria no están disponibles, y funcionan esencialmente en cualquier sitio—véanse más adelante las implicaciones.

Por ejemplo, si crea un empaquetado que contiene Bash con:

```
guix pack -RR -S /mybin=bin bash
```

... puede copiar ese empaquetado a una máquina que no tiene Guix, y desde su directorio, como una usuaria normal, ejecutar:

```
tar xf pack.tar.gz
./mibin/sh
```

En ese shell, si escribe `ls /gnu/store`, notará que `/gnu/store` muestra y contiene todas las dependencias de `bash`, ¡incluso cuando la máquina no tiene el directorio `/gnu/store`! Esto es probablemente el modo más simple de desplegar software construido en Guix en una máquina no-Guix.

**Nota:** No obstante hay un punto a tener en cuenta: esta técnica descansa en la característica de *espacios de nombres de usuaria* del núcleo Linux, la cual permite a usuarias no privilegiadas montar o cambiar la raíz. Versiones antiguas de Linux no los implementan, y algunas distribuciones GNU/Linux los desactivan.

Para producir binarios reposicionables que funcionen incluso en ausencia de espacios de nombre de usuaria, proporcione `--relocatable` o `-R dos veces`. En ese caso, los binarios intentarán el uso de espacios de nombres de usuaria y usarán PRoot si no es posible.

El programa PRoot ([https://proot-me.github.io/](https://proot-me.github.io/)) proporciona el soporte necesario para la virtualización del sistema de ficheros. Lo consigue mediante el uso de la llamada al sistema ptrace en el programa en ejecución. Esta aproximación tiene la ventaja de funcionar sin soporte especial en el núcleo, pero incurre en una sobrecarga en el tiempo de ejecución cada vez que se realiza una llamada al sistema.

```
--entry-point=orden
```

Usa `orden` como el punto de entrada del empaquetado resultante, si el formato de empaquetado lo permite—actualmente docker y squashfs (Singularity) lo permiten. `orden` debe ser una ruta relativa al perfil contenido en el empaquetado.

El punto de entrada especifica la orden que herramientas como docker run o singularity run arrancan de manera automática de forma predeterminada. Por ejemplo, puede ejecutar:

```
guix pack -f docker --entry-point=bin/guile guile
```

El empaquetado resultante puede cargarse fácilmente y docker run sin parámetros adicionales lanzará `bin/guile`:

```
docker load -i pack.tar.gz
docker run image-id
```

```
--expression=expr
-e expr  Considera el paquete al que evalúa expr
```
Esto tiene el mismo propósito que la opción del mismo nombre en `guix build` (véase Sección 7.1.3 [Opciones de construcción adicionales], página 124).

```
--manifest=fichero
-m fichero
```
Usa los paquetes contenidos en el objeto manifest devuelto por el código Scheme en `fichero`. Esta opción puede repetirse varias veces, en cuyo caso los manifiestos se concatenan.

Esto tiene un propósito similar al de la opción del mismo nombre en `guix package` (véase [profile-manifest], página 39) y usa los mismos ficheros de manifiesto. Esto le permite definir una colección de paquetes una vez y usarla tanto para crear perfiles como para crear archivos en máquinas que no tienen instalado Guix. Fíjese que puede especificar o bien un fichero de manifiesto o bien una lista de paquetes, pero no ambas.

```
--system=sistema
-s sistema
```
Intenta construir paquetes para `sistema`—por ejemplo, `x86_64-linux`—en vez del tipo de sistema de la máquina de construcción.

```
--target=tripleta
```
Compilación cruzada para la `tripleta`, que debe ser una tripleta GNU válida, como "mips64el-linux-gnu" (véase Sección “Specifying target triplets” en Autoconf).

```
--compression=herramienta
-C herramienta
```
Comprime el archivador resultante usando `herramienta`—un valor que puede ser `gzip`, `bzip2`, `xz`, `lzip` o `none` para no usar compresión.

```
--symlink=spec
-S spec
```
Añade los enlaces simbólicos especificados por `spec` al empaquetado. Esta opción puede aparecer varias veces.

La forma de `spec` es `fuente=destino`, donde `fuente` es el enlace simbólico que será creado y `destino` es el destino del enlace simbólico.

Por ejemplo, `--symlink /opt/gnu/bin=bin` crea un enlace simbólico `/opt/gnu/bin` apuntando al subdirectorio `bin` del perfil.

```
--save-provenance
```
Almacena la información de procedencia para paquetes proporcionados en la línea de órdenes. La información de procedencia incluye la URL y revisión de los canales en uso (véase Sección 4.7 [Canales], página 54).

La información de procedencia se almacena en el fichero `/gnu/store/...-profile/manifest` dentro del empaquetado, junto a los metadatos habituales del paquete—el nombre y la versión de cada paquete, sus entradas propagadas, etcétera. Es información útil para la parte receptora del empaquetado, quien de ese modo conoce como se obtuvo (supuestamente) dicho empaquetado.

Esta opción no se usa de manera predeterminada debido a que, como las marcas de tiempo, la información de procedencia no aportan nada al proceso de
construcción. En otras palabras, hay una infinidad de URL de canales e identificadores de revisiones que pueden llevar al mismo empaquetado. Almacenar estos metadatos “silenciosos” en la salida puede potencialmente romper la propiedad de reproducibilidad bit a bit entre fuentes y binarios.

--root=fichero
-r fichero
Hace que fichero sea un enlace simbólico al empaquetado resultante, y lo registra como una raíz del recolector de basura.

--localstatedir
--profile-name=nombre
Incluye el “directorio de estado local”, /var/guix, en el empaquetado resultante, y notablemente el perfil /var/guix/profiles/per-user/root/nombre—por defecto nombre es guix-profile, que corresponde con ~root/.guix-profile.

/var/guix contiene la base de datos del almacén (véase Sección 6.4 [El almacén], página 98) así como las raíces del recolector de basura (véase Sección 4.5 [Invocación de guix gc], página 48). Proporcionarlo junto al empaquetado significa que el almacén está “completo” y Guix puede trabajar con él; no proporcionarlo significa que el almacén está “muerto”: no se pueden añadir o borrar nuevos elementos después de la extracción del empaquetado.

Un caso de uso para esto es el archivador tar autocontenido de binarios de Guix (véase Sección 2.1 [Instalación binaria], página 5).

--derivation
-d
Imprime el nombre de la derivación que construye el empaquetado.

--bootstrap
Usa los binarios del lanzamiento para construir el empaquetado. Esta opción es útil únicamente a las desarrolladoras de Guix.

Además, guix pack acepta todas las opciones comunes de construcción (véase Sección 7.1.1 [Opciones comunes de construcción], página 119) y todas las opciones de transformación de paquetes (véase Sección 7.1.2 [Opciones de transformación de paquetes], página 122).
6 Interfaz programática

GNU Guix proporciona varias interfaces programáticas Scheme (APIs) para definir, construir y consultar paquetes. La primera interfaz permite a las usuarias escribir definiciones de paquetes a alto nivel. Estas definiciones referencing conceptos familiares de empaquetamiento, como el nombre y la versión de un paquete, su sistema de construcción y sus dependencias. Estas definiciones se pueden convertir en acciones concretas de construcción.

Las acciones de construcción son realizadas por el daemon Guix, en delegación de las usuarias. En una configuración estándar, el daemon tiene acceso de escritura al almacén—el directorio /gnu/store—mientras que las usuarias no. En la configuración recomendada el daemon también realiza las construcciones en chroots, bajo usuarias específicas de construcción, para minimizar la interferencia con el resto del sistema.

Las APIs de nivel más bajo están disponibles para interactuar con el daemon y el almacén. Para instruir al daemon para realizar una acción de construcción, las usuarias realmente proporcionan una derivación. Una derivación es una representación de bajo nivel de las acciones de construcción a tomar, y el entorno en el que deberían suceder—las derivaciones son a las definiciones de paquetes lo que es el ensamblador a los programas en C. El término “derivación” viene del hecho de que los resultados de la construcción derivan de ellas.

Este capítulo describe todas estas APIs en orden, empezando por las definiciones de alto nivel de paquetes.

6.1 Módulos de paquetes

Desde un punto de vista programático, las definiciones de paquetes de la distribución GNU se proporcionan por módulos Guile en el espacio de nombres (gnu packages ...) (véase Sección “Módulos” en GNU Guile Reference Manual). Por ejemplo, el módulo (gnu packages emacs) exporta una variable con nombre emacs, que está asociada a un objeto <package> (véase Sección 6.2 [Definición de paquetes], página 77).

El espacio de nombres de módulos (gnu packages ...) se recorre automáticamente en busca de paquetes en las herramientas de línea de ordenes. Por ejemplo, cuando se ejecuta guix install emacs, todos los módulos (gnu packages ...) son procesados hasta encontrar uno que exporte un objeto de paquete cuyo nombre sea emacs. Esta búsqueda de paquetes se implementa en el módulo (gnu packages).

Las usuarias pueden almacenar definiciones de paquetes en módulos con nombres diferentes—por ejemplo, (mis-paquetes emacs). Existen dos maneras de hacer visibles estas definiciones de paquetes a las interfaces de usuaria:

1. Mediante la adición del directorio que contiene sus módulos de paquetes a la ruta de búsqueda con la opción -L de guix package y otras órdenes (véase Sección 7.1.1

---

1 Fíjese que los paquetes bajo el espacio de nombres de módulo (gnu packages ...) no son necesariamente “paquetes GNU”. Este esquema de nombrado de módulos sigue la convención habitual de Guile para el nombrado de módulos: gnu significa que estos módulos se distribuyen como parte del sistema GNU, y packages identifica módulos que definen paquetes.

2 Fíjese que el nombre de fichero y el nombre de módulo deben coincidir. Por ejemplo, el módulo (mis-paquetes emacs) debe almacenarse en el fichero mis-paquetes/emacs.scm en relación con la ruta de carga especificada con --load-path o GUIX_PACKAGE_PATH. Véase Sección “Modules and the File System” en GNU Guile Reference Manual, para obtener detalles.
[Opciones comunes de construcción], página 119), o usando la variable de entorno GUIX_PACKAGE_PATH descrita más adelante.

2. Mediante la definición de un canal y la configuración de guix pull de manera que se actualice desde él. Un canal es esencialmente un repositorio Git que contiene módulos de paquetes. Véase Sección 4.7 [Canales], página 54, para más información sobre cómo definir y usar canales.

GUIX_PACKAGE_PATH funciona de forma similar a otras variables de rutas de búsqueda:

GUIX_PACKAGE_PATH [Variable de entorno]

Es una lista separada por dos puntos de directorios en los que se buscarán módulos de paquetes adicionales. Los directorios enumerados en esta variable tienen preferencia sobre los propios módulos de la distribución.

La distribución es auto-contenida y completamente basada en el lanzamiento inicial: cada paquete se construye basado únicamente en otros paquetes de la distribución. La raíz de este grafo de dependencias es un pequeño conjunto de binarios del lanzamiento inicial, proporcionados por el módulo (gnu packages bootstrap). Para más información sobre el lanzamiento inicial, véase Capítulo 12 [Lanzamiento inicial], página 427.

6.2 Definición de paquetes

La interfaz de alto nivel de las definiciones de paquetes está implementada en los módulos (guix packages) y (guix build-system). Como un ejemplo, la definición de paquete, o receta, para el paquete GNU Hello es como sigue:

(define-module (gnu packages hello)
  #:use-module (guix packages)
  #:use-module (guix download)
  #:use-module (guix build-system gnu)
  #:use-module (guix licenses)
  #:use-module (gnu packages gawk))

(define-public hello
  (package
    (name "hello")
    (version "2.10")
    (source (origin
      (method url-fetch)
      (uri (string-append "mirror://gnu/hello/hello-" version ".tar.gz"))
      (sha256
        (base32
          "0ssi1wpaf7plaswqqjwigppsg5fyh99vdlb9kzl7c9lmg89ndq1i"))))
    (build-system gnu-build-system)
    (arguments ’(#:configure-flags ’("--enable-silent-rules"))
    (inputs ’(("gawk" ,gawk)))
    (synopsis "Hello, GNU world: An example GNU package")
    (description "Guess what GNU Hello prints!")
Sin ser una experta en Scheme—pero conociendo un poco de inglés—, la lectora puede haber supuesto el significado de varios campos aquí. Esta expresión asocia la variable hello al objeto <package>, que esencialmente es un registro (véase Sección “SRFI-9” en GNU Guile Reference Manual). Este objeto de paquete puede ser inspeccionado usando los procedimientos encontrados en el módulo (guix packages); por ejemplo, (package-name hello) devuelve—¡sorpresa!—"hello".

Con suerte, puede que sea capaz de importar parte o toda la definición del paquete de su interés de otro repositorio, usando la orden guix import (véase Sección 7.5 [Invocación de guix import], página 131).

En el ejemplo previo, hello se define en un módulo para ella, (gnu packages hello). Técnicamente, esto no es estrictamente necesario, pero es conveniente hacerlo: todos los paquetes definidos en módulos bajo (gnu packages ...) se reconocen automáticamente en las herramientas de línea de órdenes (véase Sección 6.1 [Módulos de paquetes], página 76).

Hay unos pocos puntos que merece la pena destacar de la definición de paquete previa:

- El campo source del paquete es un objeto <origin> (véase Sección 6.2.2 [Referencia de origin], página 83, para la referencia completa). Aquí se usa el método url-fetch de (guix download), lo que significa que la fuente es un fichero a descargar por FTP o HTTP.

El prefijo mirror://gnu instruye a url-fetch para usar uno de los espejos GNU definidos en (guix download).

El campo sha256 especifica el hash SHA256 esperado del fichero descargado. Es obligatorio, y permite a Guix comprobar la integridad del fichero. La forma (base32 ...) introduce la representación base32 del hash. Puede obtener esta información con guix download (véase Sección 7.3 [Invocación de guix download], página 129) y guix hash (véase Sección 7.4 [Invocación de guix hash], página 130).

Cuando sea necesario, la forma origin también puede tener un campo patches con la lista de parches a ser aplicados, y un campo snippet con una expresión Scheme para modificar el código fuente.

- El campo build-system especifica el procedimiento de construcción del paquete (véase Sección 6.3 [Sistemas de construcción], página 85). Aquí, gnu-build-system representa el familiar sistema de construcción GNU, donde los paquetes pueden configurarse, construirse e instalarse con la secuencia de órdenes habitual ./configure && make && make check && make install.

- El campo arguments especifica las opciones para el sistema de construcción (véase Sección 6.3 [Sistemas de construcción], página 85). Aquí son interpretadas por gnu-build-system como una petición de ejecutar configure con la opción --enable-silent-rules.

La secuencia almohadilla-dos puntos (#:) define una palabra clave Scheme (véase Sección “Keywords” en GNU Guile Reference Manual), y #:configure-flags es una palabra clave usada para pasar un parámetro nominal al sistema de construcción (véase Sección “Coding With Keywords” en GNU Guile Reference Manual).

- El campo inputs especifica las entradas al proceso de construcción—es decir, dependencias de tiempo de construcción o ejecución del paquete. Aquí, definimos una entrada llamada "gawk", cuyo valor es el de la variable gawk; gawk en sí apunta a un objeto <package>.

De nuevo, ‘ (un acento grave, sinónimo de quasiquote) nos permite introducir una lista literal en el campo inputs, mientras que , (una coma, sinónimo de unquote) nos permite insertar un valor en dicha lista (véase Sección “Expression Syntax” en GNU Guile Reference Manual).

Fíjese que no hace falta que GCC, Coreutils, Bash y otras herramientas esenciales se especifiquen como entradas aquí. En vez de eso, gnu-build-system se hace cargo de asegurar que están presentes (véase Sección 6.3 [Sistemas de construcción], página 85).

No obstante, cualquier otra dependencia debe ser especificada en el campo inputs. Las dependencias no especificadas aquí simplemente no estarán disponibles para el proceso de construcción, provocando posiblemente un fallo de construcción.

Véase Sección 6.2.1 [Referencia de package], página 81, para una descripción completa de los campos posibles.

Una vez la definición de paquete esté en su lugar, el paquete puede ser construido realmente usando la herramienta de línea de órdenes guix build (véase Sección 7.1 [Invocación de guix build], página 119), pudiendo resolver cualquier fallo de construcción que encuentre (véase Sección 7.1.4 [Depuración de fallos de construcción], página 127). Puede volver a la definición del paquete fácilmente usando la orden guix edit (véase Sección 7.2 [Invocación de guix edit], página 129). Véase Sección 14.4 [Guías de empaquetamiento], página 438, para más información sobre cómo probar definiciones de paquetes, y Sección 7.7 [Invocación de guix lint], página 141, para información sobre cómo comprobar la consistencia del estilo de una definición. Por último, véase Sección 4.7 [Canales], página 54, para información sobre cómo extender la distribución añadiendo sus propias definiciones de paquetes en un “canal”.

Finalmente, la actualización de la definición con una nueva versión oficial puede ser automatizada parcialmente por la orden guix refresh (véase Sección 7.6 [Invocación de guix refresh], página 136).

Tras el telón, una derivación correspondiente al objeto <package> es calculada mediante el procedimiento package-derivation. Esta derivación es almacenada en un fichero .drv bajo /gnu/store. Las acciones de construcción que prescribe pueden entonces llevarse a cabo usando el procedimiento build-derivations (véase Sección 6.4 [El almacén], página 98).

**package-derivation almacén paquete [sistema] [Procedimiento Scheme]**

Devuelve el objeto <derivation> del paquete para el sistema (véase Sección 6.5 [Derivaciones], página 101).

paquete debe ser un objeto <package> válido, y sistema debe ser una cadena que denote el tipo de sistema objetivo—por ejemplo, "x86_64-linux" para un sistema GNU x86_64 basado en Linux. almacén debe ser una conexión al daemon, que opera en el almacén (véase Sección 6.4 [El almacén], página 98).
De manera similar, es posible calcular una derivación que construye de forma cruzada un paquete para otro sistema:

```scheme
package-cross-derivation almacén paquete [sistema] plataforma
```

Devuelve el objeto `<derivation>` de `paquete` compilado de forma cruzada desde `sistema` a `plataforma`.

`plataforma` debe ser una tripleta GNU válida que identifique al hardware y el sistema operativo deseado, como por ejemplo "mips64el-linux-gnu" (véase Sección “Specifying Target Triplets” en Autoconf).

Los paquetes se pueden manipular de forma arbitraria. Un ejemplo de transformación útil es la reescritura de entradas, donde el árbol de dependencias de un paquete se reescribe reemplazando entradas específicas por otras:

```scheme
package-input-rewriting reemplazos [nombre-reescrito]
```

Devuelve un procedimiento que, cuando se le pasa un paquete, reemplaza sus dependencias directas e indirectas (pero no sus entradas implícitas) de acuerdo a `reemplazos`.

`reemplazos` es una lista de pares de paquetes; el primer elemento de cada par es el paquete a reemplazar, el segundo es el reemplazo.

Opcionalmente, `nombre-reescrito` es un procedimiento de parámetro que toma el nombre del paquete y devuelve su nuevo nombre tras la reescritura.

Considere este ejemplo:

```scheme
(define libressl-en-vez-de-openssl
  ;; Esto es un procedimiento para reemplazar OPENSSL
  ;; por LIBRESSL, recursivamente.
  (package-input-rewriting ‘((,openssl . ,libressl))))

(define git-con-libressl
  (libressl-en-vez-de-openssl git))
```

Aquí primero definimos un procedimiento de reescritura que substituye `openssl` por `libressl`. Una vez hecho esto, lo usamos para definir una variante del paquete `git` que usa `libressl` en vez de `openssl`. Esto es exactamente lo que hace la opción de línea de órdenes `--with-input` (véase Sección 7.1.2 [Opciones de transformación de paquetes], página 122).

La siguiente variante de `package-input-rewriting` puede encontrar paquetes a reemplazar por su nombre en vez de por su identidad.

```scheme
package-input-rewriting/spec reemplazos
```

Devuelve un procedimiento que, proporcionado un paquete, realiza los `reemplazos` proporcionados sobre todo el grafo del paquete (excluyendo las entradas implícitas). `reemplazos` es una lista de pares de especificación y procedimiento; cada especificación es una especificación de paquete como "gcc" o "guile@2", y cada procedimiento toma un paquete que corresponda con la especificación y devuelve un reemplazo para dicho paquete.
El ejemplo previo podría ser reescrito de esta forma:

```scheme
(define libressl-en-vez-de-openssl
 ;; Reemplaza todos los paquetes llamados "openssl" con LibreSSL.
 (package-input-rewriting/spec '(("openssl" . ,(const libressl))))
)
```

La diferencia principal en este caso es que, esta vez, los paquetes se buscan por su especificación y no por su identidad. En otras palabras, cualquier paquete en el grafo que se llame openssl será reemplazado.

Un procedimiento más genérico para reescribir el grafo de dependencias de un paquete es `package-mapping`: acepta cambios arbitrarios sobre nodos del grafo.

```scheme
package-mapping proc [cortar?]  
```

Devuelve un procedimiento que, dado un paquete, aplica `proc` a todos los paquetes de los que depende y devuelve el paquete resultante. El procedimiento para la recursión cuando `cortar?` devuelve verdadero para un paquete dado.

### 6.2.1 Referencia de `package`

Esta sección resume todas las opciones disponibles en declaraciones `package` (véase Sección 6.2 [Definición de paquetes], página 77).

`package`  

Este es el tipo de datos que representa la receta de un paquete.

- **name**: El nombre del paquete, como una cadena.
- **version**: La versión del paquete, como una cadena.
- **source**: Un objeto que determina cómo se debería obtener el código fuente del paquete. La mayor parte del tiempo, es un objeto `origin`, que denota un fichero obtenido de Internet (véase Sección 6.2.2 [Referencia de origin], página 83). También puede ser cualquier otro objeto “tipo-fichero” como `local-file`, que denota un fichero del sistema local de ficheros (véase Sección 6.7 [Expresiones-G], página 108).
- **build-system**: El sistema de construcción que debe ser usado para construir el paquete (véase Sección 6.3 [Sistemas de construcción], página 85).
- **arguments** (predeterminados: `'()`)
  
  Los parámetros que deben ser pasados al sistema de construcción. Es una lista que normalmente contiene una secuencia de pares de palabra clave y valor.
- **inputs** (predeterminadas: `'()`)
- **native-inputs** (predeterminadas: `'()`)
- **propagated-inputs** (predeterminadas: `'()`)

Estos campos enumeran las dependencias del paquete. Cada uno es una lista de tuplas, donde cada tupla tiene una etiqueta para la entrada (una cadena) como su primer elemento, un paquete, origen o derivación como su segundo elemento, y opcionalmente el nombre de la salida que debe ser usada, cuyo valor predeterminado es "out" (véase Sección 4.4 [Paquetes
con múltiples salidas], página 47, para más información sobre salidas de paquetes). Por ejemplo, la lista siguiente especifica tres entradas:

```
('("libffi",libffi)
 ("libunistring",libunistring)
 ("glib:bin",glib "bin")'); la salida "bin" de Glib
```

La distinción entre **native-inputs** y **inputs** es necesaria cuando se considera la compilación cruzada. Cuando se compila desde una arquitectura distinta, las dependencias enumeradas en **inputs** son construidas para la arquitectura **objetivo**; de modo contrario, las dependencias enumeradas en **native-inputs** se construyen para la arquitectura de la máquina de construcción.

**native-inputs** se usa típicamente para enumerar herramientas necesarias en tiempo de construcción, pero no en tiempo de ejecución, como Autoconf, Automake, pkg-config, Gettext o Bison. **guix lint** puede informar de probables errores en este área (véase Sección 7.7 [Invocación de guix lint], página 141).

Por último, **propagated-inputs** es similar a **inputs**, pero los paquetes especificados se instalarán automáticamente junto al paquete al que pertenecen (véase [package-cmd-propagated-inputs], página 37, para información sobre cómo **guix package** maneja las entradas propagadas).

Por ejemplo esto es necesario cuando una biblioteca C/C++ necesita cabeceras de otra biblioteca para compilar, o cuando un fichero pkg-config se refiere a otro **via** su campo **Requires**.

Otro ejemplo donde **propagated-inputs** es útil es en lenguajes que carecen de la facilidad de almacenar la ruta de búsqueda de tiempo de ejecución de la misma manera que el campo **RUNPATH** de los ficheros ELF; esto incluye Guile, Python, Perl y más. Para asegurarse que las bibliotecas escritas en esos lenguajes puedan encontrar en tiempo de ejecución el código de las bibliotecas de las que dependen, las dependencias de tiempo de ejecución deben enumerarse en **propagated-inputs** en vez de en **inputs**.

**outputs** (predeterminada: `'("out")`)
La lista de nombres de salidas del paquete. Véase Sección 4.4 [Paquetes con múltiples salidas], página 47, para usos típicos de salidas adicionales.

**native-search-paths** (predeterminadas: `'(())`
Una lista de objetos **search-path-specification** describiendo las variables de entorno de rutas de búsqueda respetadas por el paquete.

**replacement** (predeterminado: `1.0`)
Esto debe ser o bien `#f` o bien un objeto package que será usado como reemplazo para ese paquete. Véase Capítulo 11 [Actualizaciones de seguridad], página 425, para más detalles.

**synopsis** Una descripción en una línea del paquete.
Capítulo 6: Interfaz programática

description
Una descripción más elaborada del paquete.

license
La licencia del paquete; un valor de (guix licenses), o una lista de dichos valores.

home-page
La URL de la página principal del paquete, como una cadena.

supported-systems (predeterminados: %supported-systems)
La lista de sistemas en los que se mantiene el paquete, como cadenas de la forma arquitectura-núcleo, por ejemplo "x86_64-linux".

location (predeterminada: la localización de los fuentes de la forma package)
La localización de las fuentes del paquete. Es útil forzar su valor cuando se hereda de otro paquete, en cuyo caso este campo no se corrige automáticamente.

this-package
[Sintaxis Scheme]
Cuando se usa en el ámbito léxico de la definición de un paquete, este identificador resuelve al paquete que se está definiendo.

El ejemplo previo muestra cómo añadir un paquete como su propia entrada nativa cuando se compila de forma cruzada:

```scheme
(package
  (name "guile")
  ;; ...

  ;; Cuando se compila de forma cruzada, Guile, por ejemplo
  ;; depende de una versión nativa de sí mismo. Añádirla aquí.
  (native-inputs (if (%current-target-system)
                                '(('"self" ,this-package))
                                '())))
```

Es un error hacer referencia a this-package fuera de la definición de un paquete.

6.2.2 Referencia de origin

Esta sección resume todas las opciones disponibles en declaraciones origin (véase Sección 6.2 [Definición de paquetes], página 77).

origin
[Tipo de datos]
Este es el tipo de datos que representa un origen de código fuente.

uri
Un objeto que contiene el URI de las fuentes. El tipo de objeto depende del valor de method (véase a continuación). Por ejemplo, cuando se usa el método url-fetch de (guix download), los valores adecuados para uri son: una cadena que contiene una URL, o una lista de cadenas.

method
Un procedimiento que maneja el URI.
Algunos ejemplos son:

url-fetch de (guix download)
descarga un fichero de la URL HTTP, HTTPS o FTP especificada en el campo uri;
git-fetch de (guix git-download) clona el repositorio de control de versiones Git, y prepara la revisión especificada en el campo uri como un objeto git-reference; una referencia git-reference tiene esta forma:

\[\text{(git-reference}\]
\[\text{(url "https://git.savannah.gnu.org/git/hello.git")}\]
\[\text{(commit "v2.10")}]\]

sha256 Un vector de bytes que contiene el hash SHA-256 de las fuentes. Tipicamente la forma base32 se usa aquí para generar el vector de bytes de una cadena en base-32. Puede obtener esta información usando guix download (véase Sección 7.3 [Invocación de guix download], página 129) o guix hash (véase Sección 7.4 [Invocación de guix hash], página 130).

file-name (predeterminado: #f) El nombre de fichero bajo el que el código fuente se almacenará. Cuando este es #f, un valor predeterminado sensato se usará en la mayor parte de casos. En caso de que las fuentes se obtengan de una URL, el nombre de fichero de la URL se usará. Para copias de trabajo de sistemas de control de versiones, se recomienda proporcionar el nombre de fichero explícitamente ya que el predeterminado no es muy descriptivo.

patches (predeterminados: '()) Una lista de nombres de ficheros, orígenes u objetos tipo-fichero (véase Sección 6.7 [Expresiones-G], página 108) apuntando a parches que deben ser aplicados a las fuentes. La lista de parches debe ser incondicional. En particular, no puede depender del valor de %current-system o %current-target-system.

snippet (predeterminado: #f) Una expresión-G (véase Sección 6.7 [Expresiones-G], página 108) o expresión-S que se ejecutará en el directorio de fuentes. Esta es una forma conveniente de modificar el software, a veces más que un parche.

patch-flags (predeterminadas: '\n"-p1")') Una lista de opciones de línea de órdenes que deberían ser pasadas a la orden patch.

patch-inputs (predeterminada: #f) Paquetes o derivaciones de entrada al proceso de aplicación de los parches. Cuando es #f, se proporciona el conjunto habitual de entradas necesarias para la aplicación de parches, como GNU Patch.

modules (predeterminados: '()) Una lista de módulos Guile que debe ser cargada durante el proceso de aplicación de parches y mientras se ejecuta el código del campo snippet.

patch-guile (predeterminado: #f) El paquete Guile que debe ser usado durante la aplicación de parches. Cuando es #f se usa un valor predeterminado.
6.3 Sistemas de construcción

Cada definición de paquete especifica un sistema de construcción y parámetros para dicho sistema de construcción (véase Sección 6.2 [Definición de paquetes], página 77). Este campo build-system representa el procedimiento de construcción del paquete, así como las dependencias implícitas de dicho procedimiento de construcción.

Los sistemas de construcción son objetos <build-system>. La interfaz para crear y manipularlos se proporciona en el módulo (guix build-system), y otros módulos exportan sistemas de construcción reales.

En su implementación, los sistemas de construcción primero compilan los objetos package a objetos bag. Una bolsa (traducción de bag) es como un paquete, pero con menos ornamentos—en otras palabras, una bolsa es una representación a un nivel más bajo de un paquete, que contiene todas las entradas de dicho paquete, incluyendo algunas implícitamente añadidas por el sistema de construcción. Esta representación intermedia se compila entonces a una derivación (véase Sección 6.5 [Derivaciones], página 101).

Los sistemas de construcción aceptan una lista opcional de parámetros. En las definiciones de paquete, estos son pasados vía el campo arguments (véase Sección 6.2 [Definición de paquetes], página 77). Normalmente son parámetros con palabras clave (véase Sección “Optional Arguments” en GNU Guile Reference Manual). El valor de estos parámetros normalmente se evalúa en la capa de construcción—es decir, por un proceso Guile lanzado por el daemon (véase Sección 6.5 [Derivaciones], página 101).

El sistema de construcción principal es gnu-build-system, el cual implementa el procedimiento estándar de construcción para GNU y muchos otros paquetes. Se proporciona por el módulo (guix build-system gnu).

gnu-build-system
[Variable Scheme]
gnu-build-system representa el sistema de construcción GNU y sus variantes (véase Sección “Configuration” en GNU Coding Standards).

En resumen, los paquetes que lo usan se configuran, construyen e instalan con la habitual secuencia de órdenes./configure && make && make check && make install. En la práctica, algunos pasos adicionales son necesarios habitualmente. Todos estos pasos se dividen en fases separadas, notablemente3:

unpack Extrae el archivador tar de la fuente, y cambia el directorio actual al directorio recién extraído. Si la fuente es realmente un directorio, lo copia al árbol de construcción y entra en ese directorio.

patch-source-shebangs Sustituye secuencias “#!/” encontradas al inicio de los ficheros de fuentes para que hagan referencia a los nombres correctos de ficheros del almacén. Por ejemplo, esto cambia #!/bin/sh por #!/gnu/store/...-bash-4.3/bin/sh.

configure Ejecuta el guión configure con algunas opciones predeterminadas, como --prefix=/gnu/store/..., así como las opciones especificadas por el parámetro #:configure-flags.

---

3 Rogamos que se inspeccionen los módulos (guix build gnu-build-system) para más detalles sobre las fases de construcción.
build  Ejecuta `make` con la lista de opciones especificadas en `#:make-flags`. Si el parámetro `#:parallel-build?` es verdadero (por defecto), construye con `make -j`.

check  Ejecuta `make check`, u otro objetivo especificado con `#:test-target`, a menos que se pasase `#:tests? #f`. Si el parámetro `#:parallel-tests?` es verdadero (por defecto), ejecuta `make check -j`.

install  Ejecuta `make install` con las opciones enumeradas en `#:make-flags`.

patch-shebangs  Sustituye las secuencias “#!” en los ficheros ejecutables instalados.

strip  Extrae los símbolos de depuración de ficheros ELF (a menos que el valor de `#:strip-binaries?` sea falso), y los copia a la salida `debug` cuando esté disponible (véase Capítulo 10 [Instalación de ficheros de depuración], página 424).

El módulo del lado de construcción (`guix build gnu-build-system`) define `%standard-phases` como la lista predeterminada de fases de construcción. `%standard-phases` es una lista de pares símbolo/procedimiento, donde el procedimiento implementa realmente la fase.

La lista de fases usadas para un paquete particular se puede cambiar con el parámetro `#:phases`. Por ejemplo, pasar:

```
#:phases (modify-phases %standard-phases (delete 'configure))
```

significa que todas las fases descritas anteriormente serán usadas, excepto la fase `configure`.

Además, este sistema de construcción asegura que el entorno “estándar” para paquetes GNU está disponible. Esto incluye herramientas como GCC, libc, Coreutils, Bash, Make, Diffutils, grep y sed (vea el módulo `guix build system gnu` para una lista completa). A estas las llamamos las **entradas implícitas** de un paquete, porque las definiciones de paquete no las mencionan.

Hay definidos otros objetos `<build-system>` para implementar otras convenciones y herramientas usadas por paquetes de software libre. Heredan la mayor parte de `gnu-build-system`, y se diferencian principalmente en el conjunto de entradas implícitamente añadidas al proceso de construcción, y en la lista de fases ejecutadas. Algunos de estos sistemas de construcción se enumeran a continuación.

`ant-build-system`  [Variable Scheme]  
`guix build-system ant)` exporta esta variable. Implementa el procedimiento de construcción de paquetes Java que pueden construirse con la herramienta de construcción Ant (https://ant.apache.org/).

Añade tanto `ant` como el kit de desarrollo Java (JDK), que proporciona el paquete `icedtea`, al conjunto de entradas. Se pueden especificar paquetes diferentes con los parámetros `#:ant` y `#:jdk`, respectivamente.

Cuando el paquete original no proporciona un fichero Ant apropiado, el parámetro `#:jar-name` puede usarse para generar un fichero de construcción Ant `build.xml` mínimo con tareas para construir el archivo jar especificado. En este caso, el parámetro
#:source-dir se puede usar para especificar el subdirectorio de fuentes, con “src” como valor predeterminado.

El parámetro #:main-class puede usarse con el fichero de construcción Ant mínimo para especificar la clase main del archivo jar producido. Esto permite ejecutar el archivo jar. El parámetro #:test-include puede usarse para especificar la lista de tests junit a ejecutar. El valor predeterminado es (list "**/*Test.java"). #:test-exclude puede usarse para desactivar algunas pruebas. Su valor predeterminado es (list "**/Abstract*.java") ya que las clases abstractas no se pueden ejecutar como pruebas.

El parámetro #:build-target se puede usar para especificar la tarea Ant que debe ser ejecutada durante la fase build. Por defecto se ejecuta la tarea “jar”.

**android-ndk-build-system**

Esta variable es exportada por (guix build-system android-ndk). Implementa un procedimiento de construcción para paquetes Android NDK (kit de desarrollo nativo) usando un proceso de construcción específico de Guix.

El sistema de construcción asume que los paquetes instalan sus ficheros de interfaz pública (cabeceras) en el subdirectorio "include" de la salida "out" y sus bibliotecas en el subdirectorio "lib" de la salida "out".

También se asume que la unión de todas las dependencias de un paquete no tiene ficheros en conflicto.

En este momento no funciona la compilación cruzada - por lo que las bibliotecas y los ficheros de cabecera se asumen que son locales.

**asdf-build-system/source** [Variable Scheme]

Estas variables, exportadas por (guix build-system asdf), implementan procedimientos de construcción para paquetes Common Lisp usando “ASDF” (https://common-lisp.net/project/asdf). ASDF es una utilidad de definición de sistema para programas y bibliotecas Common Lisp.

El sistema asdf-build-system/source instala los paquetes en forma de fuentes, y puede ser cargado usando cualquier implementación common lisp, vía ASDF. Los otros, como asdf-build-system/sbcl, instalan sistemas binarios en el formato entendido por una implementación particular. Estos sistemas de construcción también pueden usarse para producir programas ejecutables, o imágenes lisp que contengan un conjunto precargado de paquetes.

El sistema de construcción usa convenciones de nombres. Para paquetes binarios, el paquete debería estar prefijado con la implementación lisp, como sbcl- para asdf-build-system/sbcl.

Adicionalmente, el paquete de fuentes correspondiente debe etiquetarse usando la misma convención que los paquetes python (vea Sección 14.4.5 [Módulos Python], página 443), usando el prefijo cl-.

Para paquetes binarios, cada sistema debe definirse como un paquete Guix. Si el campo origin de un paquete contiene varios sistemas, las variaciones del paquete...
pueden crearse para construir todos los sistemas. Los paquetes de fuentes, los cuales usan asdf-build-system/source, pueden contener varios sistemas.

Para crear programa ejecutables e imágenes, se pueden usar los procedimientos del lado de construcción build-program y build-image. Deben llamarse en la fase de construcción después de la fase create-symlinks, de modo que el sistema recién construido pueda ser usado dentro de la imagen resultante. build-program necesita una lista de expresiones Common Lisp a través del parámetro #:entry-program.

Si el sistema no está definido en su propio fichero .asd del mismo nombre, entonces se debe usar el parámetro #:asd-file para especificar el fichero en el que se define el sistema. Más allá, si el paquete define un sistema para sus pruebas en su fichero separado, se cargará antes de la ejecución de las pruebas si se especifica el parámetro #:test-asd-file. Si no se especifica, se probarán si existen los ficheros <sistema>-tests.asd, <system>-test.asd, tests.asd y test.asd.

Si por alguna razón el paquete debe ser nombrado de una forma diferente a la sugerida por las convenciones de nombres, el parámetro #:asd-system-name puede usarse para especificar el nombre del sistema.

cargo-build-system  

Automáticamente añade rustc y cargo al conjunto de entradas. Se puede especificar el uso de un paquete Rust distinto con el parámetro #:rust.

Las dependencias normales de cargo se añadirán a la definición del paquete a través del parámetro #:cargo-inputs como una lista de pares de nombre y especificación, donde la especificación puede ser un paquete o una definición de fuentes source. Tenga en cuenta que la especificación debe evaluar a una ruta o a un archivo comprimido con gzip que incluya un fichero Cargo.toml en su raíz, o será ignorado. De manera parecida, las dependencias de desarrollo de cargo deben añadirse a la definición del paquete a través del parámetro #:cargo-development-inputs.

En su fase configure, este sistema de construcción hará que cualquier fuente de entrada especificada en los parámetros #:cargo-inputs y #:cargo-development-inputs esté disponible para cargo. También borrará cualquier fichero Cargo.lock incluido para que sea recreado por cargo durante la fase build de construcción. La fase install instala los binarios que el “crate” haya definido.

copy-build-system  
This variable is exported by (guix build-system copy). It supports builds of simple packages that don’t require much compiling, mostly just moving files around.

It adds much of the gnu-build-system packages to the set of inputs. Because of this, the copy-build-system does not require all the boilerplate code often needed for the trivial-build-system.

To further simplify the file installation process, an #:install-plan argument is exposed to let the packager specify which files go where. The install plan is a list of (source target [filters]). filters are optional.

- When source matches a file or directory without trailing slash, install it to target.
• Si target tiene un trailling slash, instala source basename bajo target.
• De lo contrario, instala source como target.
• Cuando source es un directorio con un trailling slash, o cuando filters están activos, el trailling slash de target se interpreta con el mismo significado que antes.
• Sin filters, instala el contenido completo de source en target.
• Con filters entre #:include, #:include-regexp, #:exclude, #:exclude-regexp, solo se seleccionan archivos dependiendo de los filters. Cada filter está especificado por una lista de cadenas.
  • Con #:include, instala todos los archivos cuya parte final coincide con al menos un elemento de la lista.
  • Con #:include-regexp, instala todos los archivos cuya parte subruta coincide con al menos una expresión regular de la lista.
• Los filters #:exclude y #:exclude-regexp son el complemento de sus inclusion counterpart. Sin #:include flags, instala todos los archivos excepto aquellos que coincidan con los exclusion filters. Si ambos incluyentes y exclusivos estén especificados, las exclusiones se hacen encima de las incluyentes.

En todos los casos, las rutas relativas a source se preservan dentro de target.

**Ejemplos:**
- ("foo/bar" "share/my-app/"): Instalar bar en share/my-app/bar.
- ("foo/bar" "share/my-app/baz"): Instalar bar en share/my-app/baz.
- ("foo/" "share/my-app"): Instalar el contenido de foo dentro de share/my-app, p.e., instalar foo/sub/file en share/my-app/sub/file.
- ("foo/" "share/my-app" #:include ("sub/file"): Instalar solo foo/sub/file en share/my-app/sub/file.
- ("foo/sub" "share/my-app" #:include ("file"): Instalar foo/sub/file en share/my-app/file.

**clojure-build-system** [Variable Scheme]

Añade clojure, icedtea y zip al conjunto de entradas. Se pueden especificar paquetes diferentes con los parámetros #:clojure, #:jdk y #:zip, respectivamente.
Una lista de directorios de fuentes, directorios de pruebas y nombres de jar pueden especificarse con los parámetros #:source-list, #:test-list y #:jar-names, respectivamente. El directorio de compilación y la clase principal pueden especificarse con los parámetros #:compile-list y #:main-class, respectivamente. Otros parámetros se documentan más adelante.

Este sistema de construcción es una extensión de ant-build-system, pero con las siguientes fases cambiadas:

**build** Esta fase llama compile en Clojure para compilar los ficheros de fuentes y ejecuta jar para crear archivadores jar tanto de ficheros de fuentes y
compilados de acuerdo con las listas de inclusión y exclusión especificadas en #:aot-include y #:aot-exclude, respectivamente. La lista de exclusión tiene prioridad sobre la de inclusión. Estas listas consisten en símbolos que representan bibliotecas Clojure o la palabra clave especial #:all que representa todas las bibliotecas encontradas en los directorios de fuentes. El parámetro #:omit-source? determina si las fuentes deben incluirse en los archivadores jar.

**check**

Esta fase ejecuta las pruebas de acuerdo a las listas de inclusión y exclusión especificadas en #:test-include y #:test-exclude, respectivamente. Sus significados son análogos a los de #:aot-include y #:aot-exclude, excepto que la palabra clave especial #:all designa ahora a todas las bibliotecas Clojure encontradas en los directorios de pruebas. El parámetro #:tests? determina si se deben ejecutar las pruebas.

**install**

Esta fase instala todos los archivadores jar construidos previamente. Además de las previas, este sistema de construcción contiene una fase adicional:

**install-doc**

Esta fase instala todos los ficheros de nivel superior con un nombre que corresponda con %doc-regex. Una expresión regular diferente se puede especificar con el parámetro #:doc-regex. Todos los ficheros dentro (recursivamente) de los directorios de documentación especificados en #:doc-dirs se instalan también.

**cmake-build-system**

Esta variable se exporta en (guix build-system cmake). Implementa el procedimiento de construcción para paquetes que usen la herramienta de construcción CMake (https://www.cmake.org).

Automáticamente añade el paquete cmake al conjunto de entradas. El paquete usado se puede especificar con el parámetro #:cmake.

El parámetro #:configure-flags se toma como una lista de opciones a pasar a cmake. El parámetro #:build-type especifica en términos abstractos las opciones pasadas al compilador; su valor predeterminado es "RelWithDebInfo" (quiere decir “modo de entrega con información de depuración”), lo que aproximadamente significa que el código se compila con -O2 -g, lo cual es el caso predeterminado en paquetes basados en Autoconf.

**dune-build-system**

Esta variable se exporta en (guix build-system dune). Permite la construcción de paquetes mediante el uso de Dune (https://dune.build/), una herramienta de construcción para el lenguaje de programación OCaml. Se implementa como una extensión de ocaml-build-system que se describe a continuación. Como tal, se pueden proporcionar los parámetros #:ocaml y #:findlib a este sistema de construcción.

Automáticamente añade el paquete dune al conjunto de entradas. El paquete usado se puede especificar con el parámetro #:dune.
No existe una fase configure debido a que los paquetes dune no necesitan ser configurados típicamente. El parámetro #:build-flags se toma como una lista de opciones proporcionadas a la orden dune durante la construcción.

El parámetro #:jbuild? puede proporcionarse para usar la orden jbuild en vez de la más reciente dune durante la construcción de un paquete. Su valor predeterminado es #f.

El parámetro #:package puede proporcionarse para especificar un nombre de paquete, lo que resulta útil cuando un paquete contiene múltiples paquetes y únicamente quiere construir uno de ellos. Es equivalente a proporcionar el parámetro -p a dune.

go-build-system [Variable Scheme]
Esta variable se exporta en (guix build-system go). Implementa el procedimiento de construcción para paquetes Go usando los mecanismos de construcción de Go (https://golang.org/cmd/go/#hdr-Compile_packages_and_dependencies) estándares.

Se espera que la usuaria proporcione un valor para el parámetro #:import-path y, en algunos caso, #:unpack-path. La ruta de importación (https://golang.org/doc/code.html#ImportPaths) corresponde a la ruta del sistema de ficheros esperada por los guiones de construcción del paquete y los paquetes a los que hace referencia, y proporciona una forma de hacer referencia a un paquete Go únicamente. Está basado típicamente en una combinación de la URI remota del paquete de ficheros de fuente y la estructura jerárquica del sistema de ficheros. En algunos casos, necesitará desempaquetar el código fuente del paquete en una estructura de directorios diferente a la indicada en la ruta de importación, y #:unpack-path debe usarse en dichos casos.

Los paquetes que proporcionan bibliotecas Go deben instalar su código fuente en la salida de la construcción. El parámetro #:install-source?, cuyo valor por defecto es #t, controla si se instalará o no el código fuente. Puede proporcionarse #f en paquetes que proporcionan únicamente ficheros ejecutables.

glib-or-gtk-build-system [Variable Scheme]
Esta variable se exporta en (guix build-system glib-or-gtk). Está pensada para usarse con paquetes que usan GLib o GTK+.

Este sistema de construcción añade las dos fases siguientes a las definidas en gnu-build-system:

glib-or-gtk-wrap
La fase glib-or-gtk-wrap se asegura de que los programas en bin/son capaces de encontrar los “esquemas” GLib y los módulos GTK+ (https://developer.gnome.org/gtk3/stable/gtk-running.html). Esto se consigue recubriendo los programas en guiones de lanzamiento que establecen apropiadamente las variables de entorno GTK_PATH.

Es posible excluir salidas específicas del paquete del proceso de recubrimiento enumerando sus nombres en el parámetro #:glib-org-gtk-wrap-excluded-outputs. Esto es útil cuando se sabe que una salida no contiene binarios GLib o GTK+, y cuando empaquetar gratuitamente añadiría una dependencia de dicha salida en GLib y GTK+.
glib-or-gtk-compile-schemas
La fase glib-or-gtk-compile-schemas se asegura que todos los esquemas GSettings (https://developer.gnome.org/gio/stable/glib-compile-schemas.html) o GLib se compilan. La compilación la realiza el programa glib-compile-schemas. Lo proporciona el paquete glib:bin que se importa automáticamente por el sistema de construcción. El paquete glib que proporciona glib-compile-schemas puede especificarse con el parámetro #:glib.

Ambas fases se ejecutan tras la fase install.

guile-build-system
Este sistema de construcción es para paquetes Guile que consisten exclusivamente en código Scheme y son tan simples que no tienen ni siquiera un fichero Makefile, menos un guión configure. Compila código Scheme usando guild compile (véase Sección “Compilation” en GNU Guile Reference Manual) e instala los ficheros .scm y .go en el lugar correcto. También instala documentación.
Este sistema de construcción permite la compilación cruzada usando la opción --target de guild compile.
Los paquetes construidos con guile-build-system deben proporcionar un paquete Guile en su campo native-inputs.

julia-build-system
Esta variable se exporta en (guix build-system julia). Implementa el procedimiento de construcción usados por paquetes julia (https://julialang.org/), lo que esencialmente es similar a la ejecución de julia -e 'using Pkg; Pkg.add(package)', en un entorno donde JULIA_LOAD_PATH contiene las rutas de todos los paquetes julia de entrada. No se ejecutan las pruebas.
Los paquetes Julia necesitan que el campo file-name de la fuente sea el nombre real del paquete, con mayúsculas y minúsculas correctamente escritas.
Para los paquetes que tienen dependencias de bibliotecas dinámicas, puede que sea necesario que escriba manualmente el fichero /deps/deps.jl. Habitualmente es una línea de const variable = /gnu/store/biblioteca.so por cada dependencia, además de una función vacía check_deps() = nothing.
Algunos paquetes antiguos que no usan todavía Package.toml también necesitan la creación de este fichero. La función julia-create-package-toml ayuda en la creación de dicho fichero. Debe proporcionarle las salidas y las fuentes del paquete, su nombre (el mismo que en el parámetro file-name), el uuid del paquete, la versión del paquete y una lista de dependencias especificadas a partir de su nombre y su uuid.

minify-build-system
Esta variable se exporta en (guix build-system minify). Implementa un procedimiento de minificación para paquetes JavaScript simples.
Añade uglify-js al conjunto de entradas y lo utiliza para comprimir todos los ficheros JavaScript en el directorio src. Un paquete de minificación diferente puede especificarse con el parámetro #:uglify-js, pero se espera que el paquete escriba el código minificado en la salida estándar.
Cuando los ficheros JavaScript de entrada no se encuentran en el directorio `src`, el parámetro `#:javascript-files` puede usarse para especificar una lista de nombres de fichero que proporcionar al minificador.

**ocaml-build-system**  
[Variable Scheme]  
Esta variable se exporta en `(guix build-system ocaml)`. Implementa un procedimiento de construcción para paquetes OCaml (https://ocaml.org), que consiste en seleccionar el conjunto correcto de órdenes a ejecutar para cada paquete. Los paquetes OCaml pueden esperar la ejecución de muchas órdenes diferentes. Este sistema de construcción probará algunas de ellas.

Cuando el paquete tiene un fichero `setup.ml` presente en el nivel superior, se ejecuta `ocaml setup.ml -configure`, `ocaml setup.ml -build` y `ocaml setup.ml -install`. El sistema de construcción asumirá que este fichero se generó con http://oasis.forge.ocamlcore.org/ OASIS y se encargará de establecer el prefijo y la activación de las pruebas si no se desactivaron. Puede pasar opciones de configuración y construcción con `#:configure-flags` y `#:build-flags`, respectivamente. El parámetro `#:test-flags` puede usarse para cambiar el conjunto de opciones usadas para activar las pruebas. El parámetro `#:use-make?` puede usarse para ignorar este sistema en las fases de construcción e instalación.

Cuando el paquete tiene un fichero `configure`, se asume que es un guión de configuración hecho a mano que necesita un formato de parámetros diferente a los del sistema `gnu-build-system`. Puede añadir más opciones con el parámetro `#:configure-flags`.

Cuando el paquete tiene un fichero `Makefile` (o `#:use-make?` es `#t`), será usado y se pueden proporcionar más opciones para las fases de construcción y e instalación con el parámetro `#:make-flags`.

Por último, algunos paquetes no tienen estos ficheros y usan unas localizaciones de algún modo estándares para su sistema de construcción. En este caso, el sistema de construcción ejecutará `ocaml pkg/pkg.ml` o `ocaml pkg/build.ml` y se hará cargo de proporcionar la ruta del módulo `findlib` necesario. Se pueden pasar opciones adicionales con el parámetro `#:build-flags`. De la instalación se hace cargo `opam-installer`. En este caso, el paquete `opam` debe añadirse al campo `native-inputs` de la definición del paquete.

Fíjese que la mayoría de los paquetes OCaml asumen su instalación en el mismo directorio que OCaml, lo que no es el comportamiento deseado en guix. En particular, intentarán instalar ficheros `.so` en su directorio de módulos, lo que normalmente es aceptable puesto que está bajo el directorio del compilador de OCaml. No obstante, en guix estas bibliotecas no se pueden instalar ahí y se usa `CAML_LD_LIBRARY_PATH`. Esta variable apunta a `lib/ocaml/site-lib/stubslibs` y allí es donde las bibliotecas `.so` deberán instalarse.

**python-build-system**  
[Variable Scheme]  
Esta variable se exporta en `(guix build-system python)`. Implementa el procedimiento más o menos estándar de construcción usado por paquetes Python, que consiste en la ejecución de `python setup.py build` y `python setup.py install --prefix=/gnu/store/...`. 
Para que instalen programas independientes Python bajo bin/, se encarga de envolver dichos programas de modo que su variable de entorno PYTHONPATH apunte a las bibliotecas Python de las que dependen.

Se puede especificar el paquete Python usado para llevar a cabo la construcción con el parámetro #:python. Esta es habitualmente una forma de forzar la construcción de un paquete para una versión específica del intérprete Python, lo que puede ser necesario si el paquete es compatible únicamente con una versión del intérprete.

De manera predeterminada guix llama a setup.py bajo el control de setuptools de manera similar a lo realizado por pip. Algunos paquetes no son compatibles con setuptools (y pip), por lo que puede desactivar esta configuración estableciendo el parámetro #:use-setuptools a #f.

### perl-build-system

Esta variable se exporta en (guix build-system perl). Implementa el procedimiento de construcción estándar para paquetes Perl, lo que o bien consiste en la ejecución de perl Build.PL --prefix=/gnu/store/..., seguido de Build y Build install; o en la ejecución de perl Makefile.PL PREFIX=/gnu/store/..., seguida de make y make install, dependiendo de si Build.PL o Makefile.PL están presentes en la distribución del paquete. El primero tiene preferencia si existen tanto Build.PL como Makefile.PL en la distribución del paquete. Esta preferencia puede ser invertida mediante la especificación del valor #t en el parámetro #:make-maker?.

La invocación inicial de perl Makefile.PL o perl Build.PL pasa a su vez las opciones especificadas por los parámetros #:make-maker-flags o #:module-build-flags, respectivamente.

El paquete Perl usado puede especificarse con #:perl.

### qt-build-system

Esta variable se exporta en (guix build-system qt). Está pensado para usarse con aplicaciones que usen Qt o KDE.

Este sistema de construcción añade las dos fases siguientes a las definidas en cmake-build-system:

**check-setup**

La fase check-setup prepara el entorno para la ejecución de las comprobaciones usadas habitualmente por los programas de pruebas de Qt. Por ahora únicamente proporciona valor a algunas variables de entorno: QT_QPA_PLATFORM=offscreen, DBUS_FATAL_WARNINGS=0 y CTEST_OUTPUT_ON_FAILURE=1.

Esta fase se añade previamente a la fase check. Es una fase separada para facilitar el ajuste si fuese necesario.

**qt-wrap**

La fase qt-wrap busca las rutas de módulos de Qt5, las rutas de QML y algunas rutas XDG en las entradas y la salida. En caso de que alguna ruta se encuentre, todos los programas en los directorios bin/,sbin/,libexec/ y lib/libexec/ de la salida se envuelven en guiones que definen las variables de entorno necesarias.
Es posible excluir salidas específicas del paquete del proceso de recubrimiento enumerando sus nombres en el parámetro `#:qt-wrap-excluded-outputs`. Esto es útil cuando se sabe que una salida no contiene binarios que usen Qt, y cuando empaquetar gratuitamente nadiría una dependencia de dicha salida en Qt.

Ambas fases se añaden tras la fase `install`.

**r-build-system** [Variable Scheme]
Esta variable se exporta en `(guix build-system r)`. Implementa el procedimiento de construcción usados por paquetes R ([https://r-project.org](https://r-project.org)), lo que esencialmente es poco más que la ejecución de `R CMD INSTALL --library=/gnu/store/...` en un entorno donde `R_LIBS_SITE` contiene las rutas de todos los paquetes R de entrada. Las pruebas se ejecutan tras la instalación usando la función `R tools::testInstalledPackage`.

**rakudo-build-system** [Variable Scheme]

El paquete rakudo en uso puede especificarse con `rakudo`. El paquete `perl6-tap-harness` en uso durante las pruebas puede especificarse con `#:prove6` o eliminarse proporcionando `#f` al parámetro `with-prove6?`. El paquete `perl6-zef` en uso durante las pruebas e instalación puede especificarse con `#:zef` o eliminarse proporcionando `#f` al parámetro `with-zef?`.

**texlive-build-system** [Variable Scheme]
Esta variable se exporta en `(guix build-system texlive)`. Se usa para construir paquetes TeX en modo de procesamiento de lotes con el motor especificado. El sistema de construcción fija la variable `TEXINPUTS` para encontrar todos los ficheros de fuentes TeX en las entradas.

Por defecto ejecuta `luatex` en todos los ficheros que terminan en `ins`. Un motor y formato diferente puede especificarse con el parámetro `#:tex-format`. Los diferentes objetivos de construcción pueden especificarse con el parámetro `#:build-targets`, que espera una lista de nombres de fichero. El sistema de construcción añade únicamente `texlive-bin` y `texlive-latex-base` (ambos desde `(gnu packages tex)` a las entradas. Ambos pueden forzarse con los parámetros `#:texlive-bin` y `#:texlive-latex-base` respectivamente.

El parámetro `#:tex-directory` le dice al sistema de construcción dónde instalar los ficheros construidos bajo el árbol `texmf`.

**ruby-build-system** [Variable Scheme]
Esta variable se exporta en `(guix build-system ruby)`. Implementa el procedimiento de construcción de RubyGems usado por los paquetes Ruby, que implica la ejecución de `gem build` seguida de `gem install`. 
El campo `source` de un paquete que usa este sistema de construcción típicamente se refiere a un archivo gem, ya que este es el formato usado por las desarrolladoras Ruby cuando publican su software. El sistema de construcción desempaqueta el archivo gem, potencialmente parchea las fuentes, ejecuta la batería de pruebas, vuelve a empaquetar el archivo gem y lo instala. Adicionalmente, se puede hacer referencia a directorios y archivadores tar para permitir la construcción de archivos gem no publicados desde Git o un archivador tar de publicación de fuentes tradicional.

Se puede especificar el paquete Ruby usado mediante el parámetro `#:ruby`. Una lista de opciones adicionales pueden pasarse a la orden `gem` en el parámetro `#:gem-flags`.

`waf-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system waf)`. Implementa un procedimiento de construcción alrededor del guión `waf`. Las fases comunes—`configure`, `build` y `install`—se implementan pasando sus nombres como parámetros al guión `waf`.

El guión `waf` es ejecutado por el intérprete Python. El paquete Python usado para la ejecución puede ser especificado con el parámetro `#:python`.

`scons-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system scons)`. Implementa en procedimiento de construcción usado por la herramienta de construcción de software SCons. Este sistema de construcción ejecuta `scons` para construir el paquete, `scons test` para ejecutar las pruebas y después `scons install` para instalar el paquete.

Las opciones adicionales a pasar a `scons` se pueden especificar con el parámetro `#:scons-flags`. Los objetivos predeterminados de construcción (`build`) e instalación (`install`) pueden modificarse con `#:build-targets` y `#:install-targets` respectivamente. La versión de Python usada para ejecutar SCons puede especificarse seleccionando el paquete SCons apropiado con el parámetro `#:scons`.

`haskell-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system haskell)`. Implementa el procedimiento de construcción Cabal usado por paquetes Haskell, el cual implica la ejecución de `runhaskell Setup.hs configure --prefix=/gnu/store/...` y `runhaskell Setup.hs build`. En vez de instalar el paquete ejecutando `runhaskell Setup.hs install`, para evitar el intento de registro de bibliotecas en el directorio de sololectora del compilador en el almacén, el sistema de construcción usa `runhaskell Setup.hs copy`, seguido de `runhaskell Setup.hs register`. Además, el sistema de construcción genera la documentación del paquete ejecutando `runhaskell Setup.hs haddock`, a menos que se pasase `#:haddock? #f`. Parámetros opcionales de Haddock pueden proporcionarse con la ayuda del parámetro `#:haddock-flags`. Si el fichero `Setup.hs` no es encontrado, el sistema de construcción busca `Setup.lhs` a su vez.

El compilador Haskell usado puede especificarse con el parámetro `#:haskell` cuyo valor predeterminado es `ghc`.

`dub-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system dub)`. Implementa el procedimiento de construcción Dub usado por los paquetes D, que implica la ejecución de `dub build` y `dub run`. La instalación se lleva a cabo con la copia manual de los ficheros.
El compilador D usado puede ser especificado con el parámetro `#:ldc` cuyo valor predeterminado es `ldc`.

`emacs-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system emacs)`. Implementa un procedimiento de instalación similar al propio sistema de empaquetado de Emacs (véase Sección “Packages” en *The GNU Emacs Manual*).

It first creates the `package-autoloads.el` file, then it byte compiles all Emacs Lisp files. Differently from the Emacs packaging system, the Info documentation files are moved to the standard documentation directory and the `dir` file is deleted. The Elisp package files are installed directly under `share/emacs/site-lisp`.

`font-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system font)`. Implementa un procedimiento de instalación para paquetes de fuentes donde las proveedoras originales proporcionan ficheros de tipografía TrueType, OpenType, etc. precompilados que simplemente necesitan copiarse en su lugar. Copia los ficheros de tipografías a las localizaciones estándar en el directorio de salida.

`meson-build-system` [Variable Scheme]
Esta variable se exporta en `(guix build-system meson)`. Implementa el procedimiento de construcción para paquetes que usan Meson (https://mesonbuild.com) como su sistema de construcción.

Añade Meson y Ninja (https://ninja-build.org/) al conjunto de entradas, y pueden cambiarse con los parámetros `#:meson` y `#:ninja` en caso necesario. La versión de Meson predeterminada es `meson-for-build`, la cual es especial puesto que no limpia el `RUNPATH` de los binarios y bibliotecas durante la instalación.

Este sistema de construcción es una extensión de `gnu-build-system`, pero con las siguientes fases cambiadas por otras específicas para Meson:

**configure**
Esta fase ejecuta `meson` con las opciones especificadas en `#:configure-flags`. La opción `--build-type` siempre se fija a `plain` a menos que se especifique algo distinto en `#:build-type`.

**build**
Esta fase ejecuta `ninja` para construir el paquete en paralelo por defecto, pero esto puede cambiarse con `#:parallel-build`.

**check**
Esta fase ejecuta `ninja` con el objetivo especificado en `#:test-target`, cuyo valor predeterminado es "test".

**install**
Esta fase ejecuta `ninja install` y no puede cambiarse.

Aparte de estas, el sistema de ejecución también añade las siguientes fases:

**fix-runpath**
Esta fase se asegura de que todos los binarios pueden encontrar las bibliotecas que necesitan. Busca las bibliotecas necesarias en subdirectorios del paquete en construcción, y añade estas a `RUNPATH` en caso necesario. También elimina referencias a bibliotecas introducidas en la fase de construcción por `meson-for-build`, como las dependencias de las pruebas, que no se necesitan realmente para la ejecución del programa.
Capítulo 6: Interfaz programática

**glib-or-gtk-wrap**
Esta fase es la fase proporcionada por *glib-or-gtk-build-system*, y no está activa por defecto. Puede activarse con `#:glib-or-gtk`.

**glib-or-gtk-compile-schemas**
Esta fase es la fase proporcionada por *glib-or-gtk-build-system*, y no está activa por defecto. Puede activarse con `#:glib-or-gtk`.

**linux-module-build-system**
Este sistema de construcción es una extensión de *gnu-build-system*, pero con las siguientes fases cambiadas:

**configure**
Esta fase configura el entorno de modo que el Makefile del núcleo Linux pueda usarse para la construcción del módulo externo del núcleo.

**build**
Esta fase usa el Makefile del núcleo Linux para construir el módulo externo del núcleo.

**install**
Esta fase usa el Makefile del núcleo Linux para instalar el módulo externo del núcleo.

Es posible y útil especificar el núcleo Linux usado para la construcción del módulo (en la forma "arguments" de un paquete que usa *linux-module-build-system* use el parámetro `#:linux` para especificarlo).

**node-build-system**
Este sistema de construcción "trivial". Es trivial en el sentido de que no proporciona funcionalidad: no incorpora entradas implícitas y no tiene una noción de fases de construcción.

**trivial-build-system**
Este sistema de construcción necesita un parámetro `#:builder`. Este parámetro debe ser una expresión Scheme que construya la(s) salida(s) del paquete—como en `build-expression->derivation` (véase Sección 6.5 [Derivaciones], página 101).

### 6.4 El almacén

Conceptualmente, el almacén es el lugar donde se almacenan las derivaciones cuya construcción fue satisfactoria—por defecto, `/gnu/store`. Los subdirectorios en el almacén se denominan *elementos del almacén* o *rutas del almacén* en ocasiones. El almacén tiene una
base de datos asociada que contiene información como las rutas del almacén a las que referencia cada ruta del almacén, y la lista de elementos válidos del almacén—los resultados de las construcciones satisfactorias. Esta base de datos reside en `localstatedir/guix/db`, donde `localstatedir` es el directorio de estado especificado vía `--localstatedir` en tiempo de configuración, normalmente `/var`.

El almacén *siempre* es accedido a través del daemon en delegación de sus clientes (véase Sección 2.5 [Invocación de guix-daemon], página 15). Para manipular el almacén, los clientes se conectan al daemon por un socket de dominio Unix, le envían peticiones y leen el resultado—esto son llamadas a procedimientos remotos, o RPC.

**Nota:** Las usuarias nunca deben modificar ficheros directamente bajo el directorio `/gnu/store`. Esto llevaría a inconsistencias y rompería las premisas de inmutabilidad del modelo funcional de Guix (véase Capítulo 1 [Introducción], página 2).

Véase Sección 4.5 [Invocación de guix gc], página 48, para información sobre cómo comprobar la integridad del almacén e intentar recuperarse de modificaciones accidentales.

El módulo *(guix store)* proporciona procedimientos para conectarse al daemon y realizar RPCs. Estos se describen más adelante. Por defecto, `open-connection`, y por tanto todas las órdenes `guix`, se conectan al daemon local o a la URI especificada en la variable de entorno `GUIX_DAEMON_SOCKET`.

**GUIX_DAEMON_SOCKET**

[Variable de entorno]
Cuando se ha definido, el valor de esta variable debe ser un nombre de fichero o una URI designando el punto de conexión del daemon. Cuando es un nombre de fichero, denota un socket de dominio Unix al que conectarse. Además de nombres de ficheros, los esquemas de URI aceptados son:

- `file`
- `unix` Estos son equivalentes a los sockets de dominio Unix. `file:///var/guix/daemon-socket/socket` es equivalente a `/var/guix/daemon-socket/socket`.
- `guix` Estas URI denotan conexiones sobre TCP/IP, sin cifrado ni verificación de la máquina remota. La URI debe especificar el nombre de máquina y opcionalmente un número de puerto (por defecto se usa el puerto 44146):
  
  ```guix://principal.guix.example.org:1234```
  
  Esta configuración es apropiada para redes locales, como clusters, donde únicamente los nodos de confianza pueden conectarse al daemon de construcción en `principal.guix.example.org`.
  
  La opción `--listen` de `guix-daemon` puede usarse para indicarle que escuche conexiones TCP (véase Sección 2.5 [Invocación de guix-daemon], página 15).
- `ssh` These URIs allow you to connect to a remote daemon over SSH. This feature requires Guile-SSH (véase Sección 2.2 [Requisitos], página 7) and a working `guile` binary in `PATH` on the destination machine. It supports
public key and GSSAPI authentication. A typical URL might look like this:

```
ssh://carlos@guix.example.org:22
```

Como con guix copy, se tienen en cuenta los ficheros habituales de configuración del cliente OpenSSH (véase Sección 7.12 [Invocación de guix copy], página 155).

Esquemas URI adicionales pueden ser aceptados en el futuro.

**Nota:** La conexión con daemon de construcción remotos se considera experimental en 1.0.1. Por favor, contacte con nosotros para compartir cualquier problema o sugerencias que pueda tener (véase Capítulo 14 [Contribuir], página 435).

### open-connection [uri] [#:reserve-space? #t]

[Procedimiento Scheme]

Abre una conexión al daemon a través del socket de dominio Unix apuntado por `uri` (una cadena). Cuando `reserve-space?` es verdadero, le indica que reserve un poco de espacio extra en el sistema de ficheros de modo que el recolector de basura pueda operar incluso cuando el disco se llene. Devuelve un objeto servidor.

El valor por defecto de `uri` es `%default-socket-path`, que ese la ruta esperada según las opciones proporcionadas a `configure`.

### close-connection servidor

[Procedimiento Scheme]

Cierra la conexión al `servidor`.

### current-build-output-port

[Variable Scheme]

Esta variable está enlazada a un parámetro SRFI-39, que referencia al puerto donde los logs de construcción y error enviados por el daemon deben escribirse.

Los procedimientos que realizan RPCs toman todos como primer parámetro un objeto servidor.

### valid-path? servidor ruta

[Procedimiento Scheme]

Devuelve `#t` cuando `ruta` designa un elemento válido del almacén y `#f` en otro caso (un elemento no-válido puede existir en el disco pero aun así no ser válido, por ejemplo debido a que es el resultado de una construcción que se interrumpió o falló).

Una condición `&store-protocol-error` se eleva si `ruta` no contiene como prefijo el directorio del almacén (`/gnu/store`).

### add-text-to-store servidor nombre texto [referencias]

[Procedimiento Scheme]

Añade `texto` bajo el fichero `nombre` en el almacén, y devuelve su ruta en el almacén. `referencias` es la lista de rutas del almacén a las que hace referencia la ruta del almacén resultante.

### build-derivations almacén derivaciones [modo]

[Procedimiento Scheme]

Construye `derivaciones`, una lista de objetos `<derivation>`, nombres de fichero `.drv`, o pares derivación/salida, usando el `modo` especificado—(build-mode normal) en caso de omisión.
Fíjese que el módulo (guix monads) proporciona una mónada así como versiones monádicas de los procedimientos previos, con el objetivo de hacer más conveniente el trabajo con código que accede al almacén (véase Sección 6.6 [La mónada del almacén], página 104).

Esta sección actualmente está incompleta.

6.5 Derivaciones

Las acciones de construcción a bajo nivel y el entorno en el que se realizan se representan mediante derivaciones. Una derivación contiene las siguientes piezas de información:

- Las salidas de la derivación—las derivaciones producen al menos un fichero o directorio en el almacén, pero pueden producir más.
- Las entradas de las derivaciones—es decir, sus dependencias de tiempo de construcción—, que pueden ser otras derivaciones o simples ficheros en el almacén ( parches, guiones de construcción, etc.).
- El tipo de sistema objetivo de la derivación—por ejemplo, x86_64-linux.
- El nombre de fichero del guión de construcción en el almacén, junto a los parámetros que se le deben pasar.
- Una lista de variables de entorno a ser definidas.

Las derivaciones permiten a los clientes del daemon comunicar acciones de construcción al almacén. Existen en dos formas: como una representación en memoria, tanto en el lado del cliente como del daemon, y como ficheros en el almacén cuyo nombre termina en .drv—estos ficheros se conocen como rutas de derivación. Las rutas de derivación pueden pasarse al procedimiento build-derivations para realizar las acciones de construcción que prescriben (véase Sección 6.4 [El almacén], página 98).

Operaciones como la descarga de ficheros y las instantáneas de un control de versiones para las cuales el hash del contenido esperado se conoce previamente se modelan como derivaciones de salida fija. Al contrario que las derivaciones normales, las salidas de una derivación de salida fija son independientes de sus entradas—por ejemplo, la descarga del código fuente produce el mismo resultado independientemente del método de descarga y las herramientas usadas.

Las derivaciones de salida—es decir, los resultados de construcción—tienen un conjunto de referencias, del que informa la RPC references o la orden guix gc --references (véase Sección 4.5 [Invocación de guix gc], página 48). Las referencias son el conjunto de dependencias en tiempo de ejecución de los resultados de construcción. Las referencias son un subconjunto de las entradas de la derivación; el daemon de construcción calcula este subconjunto de forma automática mediante el procesado de todos los ficheros en las salidas.

El módulo (guix derivations) proporciona una representación de derivaciones como objetos Scheme, junto a procedimientos para crear y manipular de otras formas derivaciones. La primitiva de más bajo nivel para crear una derivación es el procedimiento derivation:
Capítulo 6: Interfaz programática

Procedimiento Scheme

derivation almacén nombre constructor args

[#:outputs "out") [#:hash #f] [#:hash-algo #f] [#:recursive? #f]
#:inputs () [:env-vars ()] [:system (%current-system)]
#:references-graphs #f] [#:allowed-references #f]
#:disallowed-references #f] [#:leaked-env-vars #f] [#:local-build? #f]
#:substitutable? #t] [#:properties ()]

Construye una derivación con los parámetros proporcionados, y devuelve el objeto resultado.

Cuando se proporcionan hash y hash-algo, una derivación de salida fija se crea—es decir, una cuyo resultado se conoce de antemano, como la descarga de un fichero. Si, además, recursive? es verdadero, entonces la salida fijada puede ser un fichero ejecutable o directorio y hash debe ser el hash de un archivador que contenga esta salida.

Cuando references-graphs es verdadero, debe ser una lista de pares de nombre de fichero/ruta del almacén. En ese caso, el grafo de referencias de cada ruta del almacén se exporta en el entorno de construcción del fichero correspondiente, en un formato de texto simple.

Cuando allowed-references es verdadero, debe ser una lista de elementos del almacén o salidas a las que puede hacer referencia la salida de la derivación. Del mismo modo, disallowed-references, en caso de ser verdadero, debe ser una lista de cosas a las que las salidas no pueden hacer referencia.

Cuando leaked-env-vars es verdadero, debe ser una lista de cadenas que denoten variables de entorno que se permite “escapear” del entorno del daemon al entorno de construcción. Esto es únicamente aplicable a derivaciones de salida fija—es decir, cuando hash es verdadero. El uso principal es permitir que variables como http_proxy sean pasadas a las derivaciones que descargan ficheros.

Cuando local-build? es verdadero, declara que la derivación no es una buena candidata para delegación y debe ser construida localmente (véase Sección 2.4.2 [Configuración de delegación del daemon], página 11). Este es el caso para pequeñas derivaciones donde los costes de transferencia de datos sobrepasarián los beneficios.

Cuando substitutable? es falso, declara que las sustituciones de la salida de la derivación no deben usarse (véase Sección 4.3 [Sustituciones], página 44). Esto es útil, por ejemplo, cuando se construyen paquetes que capturan detalles sobre el conjunto de instrucciones de la CPU anfitriona.

properties debe ser una lista asociada que describe “propiedades” de la derivación. Debe mantenerse tal cual, sin interpretar, en la derivación.

 Esto es un ejemplo con un guión de shell como constructor, asumiendo que almacén es una conexión abierta al daemon, bash apunta al ejecutable Bash en el almacén:

(use-modules (guix utils)
 (guix store)
 (guix derivations))

(let ((constructor ; añade el guión de Bash al almacén
 (add-text-to-store store "mi-constructor.sh"
"echo hola mundo > $out
" '(()))
(derivation almacen "foo"
  bash "(-e" ,builder)
  #:inputs '((,bash) (,constructor))
  #:env-vars '("HOME" . "/sindirectorio"))\n⇒ #<derivation /gnu/store/...-foo.drv => /gnu/store/...-foo>

Como puede suponerse, el uso directo de esta primitiva es algo enredado. Una mejor aproximación es escribir guiones de construcción en Scheme, ¡por supuesto! La mejor forma de hacerlo es escribir el código de construcción como una “expresión-G”, y pasarla a gexp->derivation. Para más información, véase Sección 6.7 [Expresiones-G], página 108.

En otros tiempos, gexp->derivation no existía y la creación de derivaciones con código de construcción escrito en Scheme se conseguía con build-expression->derivation, documentada más adelante. Este procedimiento está ahora obsoleto en favor del procedimiento gexp->derivation mucho más conveniente.

build-expression->derivation almacen nombre exp [Procedimiento Scheme]
Devuelve una derivación que ejecuta la expresión Scheme exp como un constructor para la derivación nombre. inputs debe ser una lista de tuplas (nombre ruta-drv sub-drv); cuando sub-drv se omite, se asume "out". modules es una lista de nombres de módulos Guile de la ruta actual de búsqueda a copiar en el almacen, compilados, y poner a disposición en la ruta de carga durante la ejecución de exp—por ejemplo, ((guix build utils) (guix build gnu-build-system)).

exp se evalúa en un entorno donde %outputs está asociada a una lista de pares salida/ruta, y donde %build-inputs está asociada a una lista de pares cadena/ruta-de-salida que provienen de inputs. De manera opcional, env-vars es una lista de pares de cadenas que especifican el nombre y el valor de las variables de entorno visibles al constructor. El constructor termina pasando el resultado de exp a exit; por tanto, cuando exp devuelve #f, la construcción se considera fallida.

exp se construye usando guile-for-build (una derivación). Cuando guile-for-build se omite o es #f, el valor del fluido %guile-for-build se usa en su lugar.

Véase el procedimiento derivation para el significado de references-graphs, allowed-references, disallowed-references, local-build? y substitutable?.

Aquí está un ejemplo de derivación de salida única que crea un directorio que contiene un fichero:

(let ((constructor '(let ((salida (assoc-ref %outputs "out")))
                     (mkdir salida) ; crea /gnu/store/...-goo
                     (call-with-output-file (string-append salida "/prueba")\n                       (lambda (p)
                         (display '(hola guix) p))))))
(build-expression->derivation almacen "goo" constructor))
6.6 La mónada del almacén

Los procedimientos que operan en el almacén descritos en la sección previa toman todos una conexión abierta al daemon de construcción en su primer parámetro. Aunque el modelo subyacente es funcional, tienen o bien efectos secundarios o dependen del estado actual del almacén.

Lo anterior es inconveniente: la conexión al daemon de construcción tiene que proporcionarse en todas estas funciones, haciendo imposible la composición de funciones que no toman ese parámetro con funciones que sí lo hacen. Lo último puede ser problemático: ya que las operaciones del almacén tienen efectos secundarios y/o dependen del estado externo, deben ser secuenciadas de manera adecuada.

Aquí es donde entra en juego el módulo *(guix monads)*. Este módulo proporciona un entorno para trabajar con mónadas, y una mónada particularmente útil para nuestros usos, la mónada del almacén. Las mónadas son una construcción que permite dos cosas: asociar “contexto” con valores (en nuestro caso, el contexto es el almacén), y la construcción de secuencias de computaciones (aquí computaciones incluye accesos al almacén). Los valores en una mónada—valores que transportan este contexto adicional—se llaman valores monádicos; los procedimientos que devuelven dichos valores se llaman procedimientos monádicos.

Considere este procedimiento “normal”:

```scheme
(define (enlace-sh almacen)
  ;; Devuelve una derivación que enlaza el ejecutable 'bash'.
  (let* ((drv (package-derivation store bash))
     (out (derivation->output-path drv))
     (sh (string-append out "/bin/bash"))
     (build-expression->derivation store "sh"
       '(symlink ,sh %output))))
```

Mediante el uso de *(guix monads)* y *(guix gexp)*, puede reescribirse como una función monádica:

```scheme
(define (enlace-sh)
  ;; Lo mismo, pero devuelve un valor monádico.
  (mlet %store-monad ((drv (package->derivation bash)))
    (gexp->derivation "sh"
     ~(symlink (string-append #$drv "/bin/bash"
                 #$output))))
```

Hay varias cosas a tener en cuenta en la segunda versión: el parámetro *store* ahora es implícito y es “hilado en las llamadas a los procedimientos monádicos *package->derivation* y *gexp->derivation*, y el valor monádico devuelto por *package->derivation* es asociado mediante el uso de *mlet* en vez de un simple *let*.

Al final, la llamada a *package->derivation* puede omitirse ya que tendrá lugar implícitamente, como veremos más adelante (véase Sección 6.7 [Expresiones-G], página 108):

```scheme
(define (enlace-sh)
  ;; Devuelve una derivación que enlaza el ejecutable 'bash'.
  (let* ((drv (package-derivation store bash))
     (out (derivation->output-path drv))
     (sh (string-append out "/bin/bash"))
     (build-expression->derivation store "sh"
       '(symlink ,sh %output))))
```
La ejecución del procedimiento monádico enlace-para-sh no tiene ningún efecto. Como alguien dijo una vez, “sales de una mónada como sales de un edificio en llamas: corriendo” (run en inglés). Por tanto, para salir de la mónada y obtener el efecto deseado se debe usar run-with-store:

```
(run-with-store (open-connection) (enlace-sh))
```

⇒ /gnu/store/...-enlace-para-sh

Fíjese que el módulo (guix monad-repl) extiende la sesión interactiva de Guile con nuevos “meta-comandos” para facilitar el trabajo con procedimientos monádicos: run-in-store y enter-store-monad. El primero se usa para “ejecutar” un valor monádico único a través del almacén:

```scheme
scheme@(guile-user)> ,run-in-store (package->derivation hello)
$1 = #<derivation /gnu/store/...-hello-2.9.drv => ...>
```

El último entra en un entorno interactivo recursivo, donde todos los valores devueltos se ejecutan automáticamente a través del almacén:

```scheme
scheme@(guile-user)> ,enter-store-monad
store-monad@(guile-user) [1]> (package->derivation hello)
$2 = #<derivation /gnu/store/...-hello-2.9.drv => ...>
store-monad@(guile-user) [1]> (text-file "foo" "Hello!"
$3 = "/gnu/store/...-foo"
store-monad@(guile-user) [1]> ,q
store-monad@(guile-user)>
```

Fíjese que los valores no-monádicos no pueden devolverse en el entorno interactivo store-monad.

Las formas sintácticas principales para tratar con mónadas en general se proporcionan por el módulo (guix monads) y se describen a continuación.

```scheme
with-monad mónada cuerpo ...
```

[Sintaxis Scheme]

Evalúa cualquier forma >>= o return en cuerpo como estando en mónada.

```scheme
return val
```

[Sintaxis Scheme]

Devuelve el valor monádico que encapsula val.

```scheme
 >>= mval mproc ...
```

[Sintaxis Scheme]

Asocia el valor monádico mval, pasando su “contenido” a los procedimientos monádicos mproc…⁴. Puede haber un mproc o varios, como en este ejemplo:

```
(run-with-state
  (with-monad %state-monad
    ( >>= (return 1)
        (lambda (x) (return (+ 1 x)))))
```

---

⁴ Esta operación es habitualmente conocida como “bind” (asociación), pero ese nombre denota un procedimiento no relacionado en Guile. Por tanto usamos este símbolo en cierto modo criptico heredado del lenguaje Haskell.
(lambda (x) (return (* 2 x))))

⇒ 4
⇒ un-estado

\[\text{mlet módula } ((\text{var mval}) \ldots) \text{ cuerpo } \ldots\]  [Sintaxis Scheme]
\[\text{mlet* módula } ((\text{var mval}) \ldots) \text{ cuerpo } \ldots \text{ Asocia las variables} \]  [Sintaxis Scheme]

\[\text{var a los valores monádicos mval en cuerpo, el cual es una secuencia de expresiones. Como con el operator bind, esto puede pensarse como el “desempaquetado” del valor crudo no-monádico dentro del ámbito del cuerpo. La forma (\text{var -> val}) asocia \text{var} al valor “normal” \text{val}, como en \text{let}. Las operaciones de asociación ocurren en secuencia de izquierda a derecha. La última expresión de cuerpo debe ser una expresión monádica, y su resultado se convertirá en el resultado de mlet o mlet* cuando se ejecute en la módula.}

\[\text{mlet* es a mlet lo que let* es a let (véase Sección “Local Bindings” en GNU Guile Reference Manual).}\]

\[\text{mbegin módula mexp } \ldots\]  [Sistema Scheme]
\[\text{Asocia mexp y las siguientes expresiones monádicas en secuencia, devolviendo el resultado de la última expresión. Cada expresión en la secuencia debe ser una expresión monádica.}
\[\text{Esto es similar a mlet, excepto que los valores devueltos por las expresiones monádicas se ignoran. En ese sentido el funcionamiento es análogo a begin pero aplicado a expresiones monádicas.}\]

\[\text{mwhen condición mexp0 mexp* } \ldots\]  [Sistema Scheme]
\[\text{Cuando condición es verdadero, evalúa la secuencia de expresiones monádicas mexp0..mexp* como dentro de mbegin. Cuando condición es falso, devuelve *unespecified* en la módula actual. Todas las expresiones en la secuencia deben ser expresiones monádicas.}\]

\[\text{munless condición mexp0 mexp* } \ldots\]  [Sistema Scheme]
\[\text{Cuando condición es falso, evalúa la secuencia de expresiones monádicas mexp0..mexp* como dentro de mbegin. Cuando condición es verdadero, devuelve *unespecified* en la módula actual. Todas las expresiones en la secuencia deben ser expresiones monádicas.}\]

El módulo (guix monads) proporciona la módula de estado, que permite que un valor adicional—el estado—sea hilado a través de las llamadas a procedimientos monádicos.

\[\%\text{state-monad}\]  [Variable Scheme]
\[\text{La módula de estado. Procedimientos en la módula de estado pueden acceder y cambiar el estado hilado.}
\[\text{Considere el siguiente ejemplo. El procedimiento cuadrado devuelve un valor en la módula de estado.}\]

\[(\text{define (cuadrado x)}\]
\[\text{(mlet \%\text{state-monad} ((\text{count (current-state))}})\]
(mbegin %state-monad
  (set-current-state (+ 1 count))
  (return (* x x))))

(run-with-state (sequence %state-monad (map cuadrado (iota 3))) 0)
⇒ (0 1 4)
⇒ 3

Cuando se “ejecuta” a través de %state-monad, obtenemos un valor adicional de estado, que ese el número de llamadas a cuadrado.

### current-state
[Procedimiento monádico]
Devuelve el estado actual como un valor monádico.

### set-current-state valor
[Procedimiento monádico]
Establece el estado actual a valor y devuelve el estado previo como un valor monádico.

### state-push valor
[Procedimiento monádico]
Apila valor al estado actual, que se asume que es una lista, y devuelve el estado previo como un valor monádico.

### state-pop
[Procedimiento monádico]
Extrae un valor del estado actual y lo devuelve como un valor monádico. Se asume que el estado es una lista.

### run-with-state mval [estado]
[Procedimiento Scheme]
Ejecuta un valor monádico mval comenzando con estado como el estado inicial. Devuelve dos valores: el valor resultante y el estado resultante.

La interfaz principal a la mónada del almacén, proporcionada por el módulo (guix store), es como sigue.

### %store-monad
[Variable Scheme]
La mónada del almacén—un alias para %state-monad.

Los valores en la mónada del almacén encapsulan los accesos al almacén. Cuando su efecto es necesario, un valor de la mónada del almacén será “evaluado” cuando se proporcione al procedimiento run-with-store (véase a continuación).

### run-with-store almacén mval [#:guile-for-build] [#:system (%current-system)]
[Procedimiento Scheme]
Ejecuta mval, un valor monádico en la mónada del almacén, en almacén, una conexión abierta al almacén.

### text-file nombre texto [referencias]
[Procedimiento monádico]
Devuelve como un valor monádico el nombre absoluto del fichero en el almacén del fichero que contiene texto, una cadena. referencias es una lista de elementos del almacén a los que el fichero de texto referencia; su valor predeterminado es la lista vacía.

### binary-file nombre datos [referencias]
[Procedimiento monádico]
Devuelve como un valor monádico el nombre absoluto del fichero en el almacén del fichero que contiene datos, un vector de bytes. referencias es una lista de elementos del almacén a los que el fichero binario referencia; su valor predeterminado es la lista vacía.
Devolvemos el nombre del fichero una vez internado en el almacén. Usa 
*nombre* como su nombre del almacén, o el nombre base de *fichero* si
*nombre* se omite.

Cuando *recursive*? es verdadero, los contenidos del *fichero* se añaden recursivamente;
si *fichero* designa un fichero plano y *recursive*? es verdadero, sus contenidos se añaden,
y sus bits de permisos se mantienen.

Cuando *recursive*? es verdadero, llama a (*select?* *fichero* *stat*) por cada entrada
del directorio, donde *fichero* es el nombre absoluto de fichero de la entrada y *stat* es
el resultado de *lstat*; excluyendo las entradas para las cuales *select?* no devuelve
verdadero.

El ejemplo siguiente añade un fichero al almacén, bajo dos nombres diferentes:

```scheme
(run-with-store (open-connection)
    (mlet %store-monad ((a (interned-file "README"))
            (b (interned-file "README" "LEGU-MIN")))
        (return (list a b))))
⇒ ("/gnu/store/rwm...-README" "/gnu/store/44i...-LEGU-MIN")
```

El módulo (*guix packages*) exporta los siguientes procedimientos monádicos relaciona-
dos con paquetes:

**package-file** *paquete* [*fichero*] [*#:system*] [*#:target*] [*#:output*]

Devuelve como un valor monádico el nombre absoluto de fichero de *fichero* dentro del
directorio de salida *output* del *paquete*. Cuando se omite *fichero*, devuelve el nombre
del directorio de salida *output* del *paquete*. Cuando *target* es verdadero, se usa como
una tripleta de compilación cruzada.

**package->derivation** *paquete* [*sistema*]

**package->cross-derivation** *paquete objetivo* [*sistema*]

Versión monádica de *package-derivation* y *package-cross-derivation* (véase
Sección 6.2 [Definición de paquetes], página 77).

### 6.7 Expresiones-G

Por tanto tenemos “derivaciones”, que representan una secuencia de acciones de construc-
ción a realizar para producir un elemento en el almacén (véase Sección 6.5 [Derivaciones],
página 101). Estas acciones de construcción se llevan a cabo cuando se solicita al dae-
mon construir realmente la derivación; se ejecutan por el daemon en un contenedor (véase
Sección 2.5 [Invocación de guix-daemon], página 15).

No debería ser ninguna sorpresa que nos guste escribir estas acciones de construcción en
Scheme. Cuando lo hacemos, terminamos con dos *estratos* de código Scheme:

---

5 El término *estrato* en este contexto se debe a Manuel Serrano et al. en el contexto de su trabajo en Hop. Oleg Kiselyov, quien ha escrito profundos ensayos sobre el tema (http://okmij.org/ftp/meta-programming/#meta-scheme), se refiere a este tipo de generación de código como separación en etapas o *staging*.
---
Capítulo 6: Interfaz programática

anfitrión”—código que define paquetes, habla al daemon, etc.—y el “código de construcción”—código que realmente realiza las acciones de construcción, como la creación de directorios, la invocación de make, etc.

Para describir una derivación y sus acciones de construcción, típicamente se necesita embeber código de construcción dentro del código anfitrión. Se resume en la manipulación de código de construcción como datos, y la homoiconicidad de Scheme—el código tiene representación directa como datos—es útil para ello. Pero necesitamos más que el mecanismo normal de quasiquote en Scheme para construir expresiones de construcción.

El módulo (guix gexp) implementa las expresiones-G, una forma de expresiones-S adaptada para expresiones de construcción. Las expresiones-G, o gexps, consiste esencialmente en tres formas sintácticas: gexp, unexp y unexp-splicing (o simplemente: #~, #$ y #$@), que son comparables a quasiquote, unquote y unquote-splicing, respectivamente (véase Sección “Expression Syntax” en GNU Guile Reference Manual). No obstante, hay importantes diferencias:

• Las expresiones-G están destinadas a escribirse en un fichero y ser ejecutadas o manipuladas por otros procesos.
• Cuando un objeto de alto nivel como un paquete o una derivación se expande dentro de una expresión-G, el resultado es el mismo que la introducción de su nombre de fichero de salida.
• Las expresiones-G transportan información acerca de los paquetes o derivaciones que representan, y estas referencias se añaden automáticamente como entradas al proceso de construcción que las usa.

Este mecanismo no se limita a objetos de paquete ni derivación: pueden definirse compiladores capaces de “bajar el nivel” de otros objetos de alto nivel a derivaciones o ficheros en el almacén, de modo que esos objetos puedan introducirse también en expresiones-G. Por ejemplo, un tipo útil de objetos de alto nivel que pueden insertarse en una expresión-G son los “objetos tipo-fichero”, los cuales facilitan la adición de ficheros al almacén y su referencia en derivaciones y demás (vea local-file y plain-file más adelante).

Para ilustrar la idea, aquí está un ejemplo de expresión-G:

(define exp-construcccion
  #~(begin
      (mkdir #$output)
      (chdir #$output)
      (symlink (string-append #$coreutils "/bin/ls")
        "enumera-ficheros"))

Esta expresión-G puede pasarse a gexp->derivation: obtenemos una derivación que construye un directorio que contiene exactamente un enlace simbólico a /gnu/store/...-coreutils-8.22/bin/ls:

(gexp->derivation "la-cosa" exp-construcccion)

Como se puede esperar, la cadena "/gnu/store/...-coreutils-8.22" se sustituye por la referencia al paquete coreutils en el código de construcción real, y coreutils se marca automáticamente como una entrada a la derivación. Del mismo modo, #$output (equivalente a (ungexp output)) se reemplaza por una cadena que contiene el nombre del directorio de la salida de la derivación.
En un contexto de compilación cruzada, es útil distinguir entre referencias a construcciones *nativas* del paquete—que pueden ejecutarse en el sistema anfitrión—de referencias de compilaciones cruzadas de un paquete. Para dicho fin, `#+` tiene el mismo papel que `#$`, pero es una referencia a una construcción nativa del paquete:

```scheme
(gexp->derivation "vi"
  "begin
    (mkdir #$output)
    (mkdir (string-append #$output "/bin"))
    (system* (string-append #+coreutils "/bin/ln")
      "-s"
      (string-append #$emacs "/bin/emacs")
      (string-append #$output "/bin/vi")))
#:target "mips64el-linux-gnu")
```

En el ejemplo previo, se usa la construcción nativa de *coreutils*, de modo que *ln* pueda realmente ejecutarse en el anfitrión; pero se hace referencia a la construcción de compilación cruzada de *emacs*.

Otra característica de las expresiones-G son los *módulos importados*: a veces deseará ser capaz de usar determinados módulos Guile del “entorno anfitrión” en la expresión-G, de modo que esos módulos deban ser importados en el “entorno de construcción”. La forma `with-imported-modules` le permite expresarlo:

```scheme
(let ((build (with-imported-modules '((guix build utils)
              "begin
              (use-modules (guix build utils))
              (mkdir-p (string-append #$output "/bin"))))))
(gexp->derivation "directorio-vacio"
  "begin
    #$build
    (display "éxito!\n")
    #t)))
```

En este ejemplo, el módulo (*guix build utils*) se incorpora automáticamente dentro del entorno de construcción aislado de nuestra expresión-G, de modo que (*use-modules (guix build utils)*) funciona como se espera.

De manera habitual deseará que la *clausura* del módulo se importe—es decir, el módulo en sí y todos los módulos de los que depende—in vez del módulo únicamente; si no se hace, cualquier intento de uso del módulo fallará porque faltan módulos dependientes. El procedimiento `source-module-closure` computa la clausura de un módulo mirando en las cabeceras de sus ficheros de fuentes, lo que es útil en este caso:

```scheme
(use-modules (guix modules)) ;para 'source-module-closure'

(with-imported-modules (source-module-closure
    '((guix build utils)
      (gnu build vm)))
(gexp->derivation "algo-con-maq-virtuales"
  "begin
    (use-modules (guix build utils)
```
Capítulo 6: Interfaz programática

(gnu build vm))

...)))

De la misma manera, a veces deseará importar no únicamente módulos puros de Scheme, pero también “extensiones” como enlaces Guile a bibliotecas C u otros paquetes “completos”. Si, digamos, necesitase el paquete guile-json disponible en el lado de construcción, esta sería la forma de hacerlo:

(use-modules (gnu packages guile)) ;para 'guile-json'
(with-extensions (list guile-json)
  (gexp->derivation "algo-con-json"
    "(begin
      (use-modules (json))
      ...)))

La forma sintáctica para construir expresiones-G se resume a continuación.

#~exp [Sintaxis Scheme]
(gexp exp)

Devuelve una expresión-G que contiene exp. exp puede contener una o más de las siguientes formas:

#$obj
(ungexp obj)

Introduce una referencia a obj. obj puede tener uno de los tipos permitidos, por ejemplo un paquete o derivación, en cuyo caso la forma unexp se substituye por el nombre de fichero de su salida—por ejemplo, `/gnu/store/...-coreutils-8.22.

Si obj es una lista, se recorre y las referencias a objetos permitidos se sustituyen de manera similar.

Si obj es otra expresión-G, su contenido se inserta y sus dependencias se añaden a aquellas de la expresión-G que la contiene.

Si obj es otro tipo de objeto, se inserta tal cual es.

#$obj:salida
(ungexp obj salida)

Como la forma previa, pero referenciando explícitamente la salida de obj—esto es útil cuando obj produce múltiples salidas (véase Sección 4.4 [Paquetes con múltiples salidas], página 47).

#+obj
#+obj:salida
(ungexp-native obj)
(ungexp-native obj salida)

Igual que unexp, pero produce una referencia a la construcción nativa de obj cuando se usa en un contexto de compilación cruzada.

#$output[:salida]
(ungexp output [salida])

Inserta una referencia a la salida de la derivación salida, o a la salida principal cuando salida se omite.
Esto únicamente tiene sentido para expresiones-G pasadas a gexp->derivation.

```scheme
#$@
(lst (ungexp-splicing lst))
Lo mismo que la forma previa, pero expande el contenido de la lista lst como parte de la lista que la contiene.
```

```scheme
#+@
(lst (ungexp-native-splicing lst))
Lo mismo que la forma previa, pero hace referencia a las construcciones nativas de los objetos listados en lst.
```

Las expresiones-G creadas por gexp o #~ son objetos del tipo gexp? en tiempo de ejecución (vea más adelante).

**with-imported-modules módulos cuerpo...**

[Sintaxis Scheme]
Marca las expresiones-G definidas en el cuerpo... como si requiriesen módulos en su entorno de ejecución.

Cada elemento en módulos puede ser el nombre de un módulo, como (guix build utils), o puede ser el nombre de un módulo, seguido de una flecha, seguido de un objeto tipo-fichero:
```
`((guix build utils)
 (guix gcrypt)
 ((guix config) => ,(scheme-file "config.scm"
 #~(define-module ...)))
```

En el ejemplo previo, los dos primeros módulos se toman de la ruta de búsqueda, y el último se crea desde el objeto tipo-fichero proporcionado.

Esta forma tiene ámbito léxico: tiene efecto en las expresiones-G definidas en cuerpo..., pero no en aquellas definidas, digamos, en procedimientos llamados por cuerpo....

**with-extensions extensiones cuerpo...**

[Sintaxis Scheme]
Marca que las expresiones definidas en cuerpo... requieren extensiones en su entorno de construcción y ejecución. extensiones es típicamente una lista de objetos de paquetes como los que se definen en el módulo (gnu packages guile).

De manera concreta, los paquetes listados en extensiones se añaden a la ruta de carga mientras se compilan los módulos importados en cuerpo...; también se añaden a la ruta de carga en la expresión-G devuelta por cuerpo....

**gexp? obj**

[Procedimiento Scheme]
Devuelve #t si obj es una expresión-G.

Las expresiones-G están destinadas a escribirse en disco, tanto en código que construye alguna derivación, como en ficheros planos en el almacén. Los procedimientos monádicos siguientes le permiten hacerlo (véase Sección 6.6 [La mónada del almacén], página 104, para más información sobre mónadas).
Devuelve una derivación nombre que ejecuta exp (una expresión-G) con guile-for-build (una derivación) en el sistema system; exp se almacena en un fichero llamado script-name. Cuando target es verdadero, se usa como la tripleta de compilación cruzada para paquetes a los que haga referencia exp.

modules está obsoleto en favor de with-imported-modules. Su significado es hacer que los módulos modules estén disponibles en el contexto de evaluación de exp; modules es una lista de nombres de módulos Guile buscados en module-path para ser copiados al almacén, compilados y disponibles en la ruta de carga durante la ejecución de exp—por ejemplo, ((guix build utils) (gui build gnu-build-system)).

effective-version determina la cadena usada cuando se añaden las extensiones de exp (vea with-extensions) a la ruta de búsqueda—por ejemplo, "2.2".

graft? determina si los paquetes a los que exp hace referencia deben ser injertados cuando sea posible.

Cuando references-graphs es verdadero, debe ser una lista de tuplas de una de las formas siguientes:

- (nombre-fichero paquete)
- (nombre-fichero paquete salida)
- (nombre-fichero derivación)
- (nombre-fichero derivación salida)
- (nombre-fichero elemento-almacén)

El lado derecho de cada elemento de references-graphs se convierte automáticamente en una entrada del proceso de construcción de exp. En el entorno de construcción, cada nombre-fichero contiene el grafo de referencias del elemento correspondiente, en un formato de texto simple.

allowed-references debe ser o bien #f o una lista de nombres y paquetes de salida. En el último caso, la lista denota elementos del almacén a los que el resultado puede hacer referencia. Cualquier referencia a otro elemento del almacén produce un error de construcción. De igual manera con disallowed-references, que enumera elementos a los que las salidas no deben hacer referencia.

deprecation-warnings determina si mostrar avisos de obsolescencia durante la compilación de los módulos. Puede ser #f, #t o 'detailed.

El resto de parámetros funcionan como en derivation (véase Sección 6.5 [Derivaciones], página 101).

Los procedimientos local-file, plain-file, computed-file, program-file y scheme-file a continuación devuelven objetos tipo-fichero. Esto es, cuando se expanden
en una expresión-G, estos objetos dirigen a un fichero en el almacén. Considere esta expresión-G:

```gexp
#~(system* #$\{$file-append glibc "/sbin/nscd" \"-f\"
#$\{$local-file "/tmp/mi-nscd.conf\"})
```

El efecto aquí es el “internamiento” de `/tmp/mi-nscd.conf` mediante su copia al almacén. Una vez expandida, por ejemplo vía `gexp->derivation`, la expresión-G hace referencia a la copia bajo `/gnu/store`; por tanto, la modificación o el borrado del fichero en `/tmp` no tiene ningún efecto en lo que la expresión-G hace. `plain-file` puede usarse de manera similar; se diferencia en que el contenido del fichero se proporciona directamente como una cadena.

**local-file fichero [nombre] [#:recursive? #f] [#:select? (const #t)]**

Elemento que representa un fichero en el almacén; este objeto puede usarse en una expresión-G. Si `fichero` es un nombre de fichero relativo, se busca de forma relativa al fichero fuente donde esta forma aparece; si `fichero` no es una cadena literal, se buscará de manera relativa al directorio de trabajo durante la ejecución. `fichero` se añadirá al almacén bajo `nombre`—de manera predeterminada el nombre de `fichero` sin los directorios.

Cuando `recursive?` es verdadero, los contenidos del `fichero` se añaden recursivamente; si `fichero` designa un fichero plano y `recursive?` es verdadero, sus contenidos se añaden, y sus bits de permisos se mantienen.

Cuando `recursive?` es verdadero, llama a `(select? fichero stat)` por cada entrada del directorio, donde `fichero` es el nombre absoluto de fichero de la entrada y `stat` es el resultado de `lstat`; excluyendo las entradas para las cuales `select?` no devuelve verdadero.

Esta es la contraparte declarativa del procedimiento monádico `interned-file` (véase Sección 6.6 [La mónada del almacén], página 104).

**plain-file nombre contenido**

Elemento que representa un fichero de texto llamado `nombre` con el contenido proporcionado (una cadena o un vector de bytes) para ser añadido al almacén.

Esta es la contraparte declarativa de `text-file`.

**computed-file nombre gexp [#:options '(#:local-build? #f)]**

El resultado del almacén que representa el elemento del almacén `nombre`, un fichero o un directorio computado por `gexp.options` es una lista de parámetros adicionales a pasar a `gexp->derivation`.

Esta es la contraparte declarativa de `gexp->derivation`.

**gexp->script nombre exp [#:guile (default-guile)] [#:module-path %load-path] [#:system (%current-system)] [#:target #f]**

Devuelve un guión ejecutable `nombre` que ejecuta `exp` usando `guile`, con los módulos importados por `exp` en su ruta de búsqueda. Busca los módulos de `exp` en `module-path`.
El ejemplo siguiente construye un guión que simplemente invoca la orden `ls`:

```scheme
(use-modules (guix gexp) (gnu packages base))

(gexp->script "enumera-ficheros"
  #(execl #$((file-append coreutils "/bin/ls")
    "ls"))
```

Cuando se ejecuta a través del almacén (véase Sección 6.6 [La mónada del almacén], página 104), obtenemos una derivación que produce un fichero ejecutable `/gnu/store/...-enumera-ficheros` más o menos así:

```scheme
#!/gnu/store/...-guile-2.0.11/bin/guile -ds
!(
  (execl "/gnu/store/...-coreutils-8.22"/bin/ls" "ls")
)
```

**Programa de fichero** `nombre exp` [#:guile #:module-path #:load-path] [Procedimiento Scheme]

Devuelve un objeto que representa el elemento ejecutable del almacén `nombre` que ejecuta `exp`. `guile` es el paquete Guile usado para ejecutar el guión. Los módulos importados por `exp` se buscan en `module-path`.

Esta es la contraparte declarativa de `gexp->script`.

**Procedimiento monádico** `gexp->file nombre exp` [#:set-load-path? #:splice? #:module-path #:splice? #:guile (default-guile)]

Devuelve una derivación que construye un fichero `nombre` que contiene `exp`. Cuando `splice?` es verdadero, se considera que `exp` es una lista de expresiones que deben ser expandidas en el fichero resultante.

Cuando `set-load-path` es verdadero, emite código en el fichero resultante para establecer `%load-path` y `%load-compiled-path` de manera que respeten los módulos importados por `exp`. Busca los módulos de `exp` en `module-path`.

El fichero resultante hace referencia a todas las dependencias de `exp` o a un subconjunto de ellas.

**Procedimiento Scheme** `scheme-file nombre exp` [#:splice? #:splice?] [Procedimiento Scheme]

Devuelve un objeto que representa el fichero Scheme `nombre` que contiene `exp`.

Esta es la contraparte declarativa de `gexp->file`.

**Procedimiento monádico** `text-file* nombre texto ...` [Procedimiento monádico]

Devuelve como un valor monádico una derivación que construye un fichero de texto que contiene todo `texto`. `texto` puede ser una lista de, además de cadenas, objetos de cualquier tipo que pueda ser usado en expresiones-G: paquetes, derivaciones, ficheros locales, objetos, etc. El fichero del almacén resultante hace referencia a todos ellos.

Esta variante debe preferirse sobre `text-file` cuando el fichero a crear haga referencia a elementos del almacén. Esto es el caso típico cuando se construye un fichero de configuración que embebe nombres de ficheros del almacén, como este:

```scheme
(define (perfil.sh)
  ;; Devuelve el nombre de un guión shell en el almacén
  ;; que establece la variable de entorno 'PATH'
```
En este ejemplo, el fichero /gnu/store/...-perfil.sh resultante hará referencia a coreutils, grep y sed, por tanto evitando que se recolecten como basura durante su tiempo de vida.

mixed-text-file nombre texto ... [Procedimiento Scheme]
Devuelve un objeto que representa el fichero del almacén nombre que contiene texto. texto es una secuencia de cadenas y objetos tipo-fichero, como en:

(mixed-text-file "perfil"
 "export PATH=" coreutils "/bin:
 grep "/bin:
 sed "/bin\n")

Esta es la contraparte declarativa de text-file*.

file-union nombre ficheros [Procedimiento Scheme]
Devuelve un <computed-file> que construye un directorio que contiene todos los ficheros. Cada elemento en ficheros debe ser una lista de dos elementos donde el primer elemento es el nombre de fichero usado en el nuevo directorio y el segundo elemento es una expresión-G que denota el fichero de destino. Aquí está un ejemplo:

(file-union "etc"
 ' ("hosts")
 ("bashrc")
)

Esto emite un directorio etc que contiene estos dos ficheros.

directory-union nombre cosas [Procedimiento Scheme]
Devuelve un directorio que es la unión de cosas, donde cosas es una lista de objetos tipo-fichero que denotan directorios. Por ejemplo:

(directory-union "guile+emacs" (list guile emacs))

emite un directorio que es la unión de los paquetes guile y emacs.

file-append obj sufijo ... [Procedimientos Scheme]
Devuelve un objeto tipo-fichero que se expande a la concatenación de obj y sufijo, donde obj es un objeto que se puede bajar de nivel y cada sufijo es una cadena.

Como un ejemplo, considere esta expresión-G:

(gexp->script "ejecuta-uname"
 #"(system* #$($file-append coreutils "/bin/uname")))

El mismo efecto podría conseguirse con:

(gexp->script "ejecuta-uname"
 #"(system* $(string-append #$coreutils "/bin/uname")))

Hay una diferencia no obstante: en el caso de file-append, el guión resultante contiene una ruta absoluta de fichero como una cadena, mientras que en el segundo caso, el guión resultante contiene una expresión (string-append ...) para construir el nombre de fichero en tiempo de ejecución.
with-parameters ((parameter value) ...) exp

This macro is similar to the parameterize form for dynamically-bound parameters (véase Sección “Parameters” en GNU Guile Reference Manual). The key difference is that it takes effect when the file-like object returned by exp is lowered to a derivation or store item.

A typical use of with-parameters is to force the system in effect for a given object:

(with-parameters ((%current-system "i686-linux"))
  coreutils)

The example above returns an object that corresponds to the i686 build of Coreutils, regardless of the current value of %current-system.

Por supuesto, además de expresiones-G embebidas en código “anfitrión”, hay también módulos que contienen herramientas de construcción. Para clarificar que están destinados para su uso en el estrato de construcción, estos módulos se mantienen en el espacio de nombres (guix build ...).

Internamente, los objetos de alto nivel se bajan de nivel, usando su compilador, a derivaciones o elementos del almacén. Por ejemplo, bajar de nivel un paquete emite una derivación, y bajar de nivel un plain-file emite un elemento del almacén. Esto se consigue usando el procedimiento monádico lower-object.

lower-object obj [sistema] [#:target #f]

Devuelve como un valor en %store-monad la derivación o elemento del almacén que corresponde a obj en sistema, compilando de manera cruzada para target si target es verdadero. obj debe ser un objeto que tiene asociado un compilador de expresiones-G, como <package>.

6.8 Invocación de guix repl

La orden guix repl lanza un bucle de lectura-evaluación-impresión Guile (REPL) para programación interactiva (véase Sección “Using Guile Interactively” en GNU Guile Reference Manual). Comparado a simplemente lanzar la orden guile, guix repl garantiza que todos los módulos Guix y todas sus dependencias están disponibles en la ruta de búsqueda. Puede usarla de esta manera:

$ guix repl
  scheme@(guile-user)> ,use (gnu packages base)
  scheme@(guile-user)> coreutils
  $1 = #<package coreutils@8.29 gnu/packages/base.scm:327 3e28300>

Además, guix repl implementa un protocolo del REPL simple legible por máquinas para su uso por (guix inferior), una facilidad para interactuar con inferiores, procesos separados que ejecutan una revisión de Guix potencialmente distinta.

Las opciones disponibles son las siguientes:

--type=tipo
-t tipo   Inicia un REPL del TIPO dado, que puede ser uno de los siguientes:

  guile   Es el predeterminado, y lanza una sesión interactiva Guile estándar con todas las características.
Lanza un REPL que usa el protocolo legible por máquinas. Este es el protocolo con el que el módulo (guix inferior) se comunica.

--listen=destino
Por defecto, guix repl lee de la entrada estándar y escribe en la salida estándar. Cuando se pasa esta opción, en vez de eso escuchará las conexiones en destino. Estos son ejemplos de opciones válidas:

--listen=tcp:37146
Acepta conexiones locales por el puerto 37146.

--listen=unix:/tmp/socket
Acepta conexiones a través del socket de dominio Unix /tmp/socket.

--load-path=directorio
-L directorio
Añade directorio al frente de la ruta de búsqueda de módulos de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76). Esto permite a las usuarias definir sus propios paquetes y hacerlos visibles a las herramientas de línea de órdenes.

-q
Inhibe la carga del fichero ~/.guile. De manera predeterminada, dicho fichero de configuración se carga al lanzar una sesión interactiva de guile.
7 Utilidades

Esta sección describe las utilidades de línea de órdenes de Guix. Algunas de ellas están orientadas principalmente para desarrolladoras y usuarias que escriban definiciones de paquetes nuevas, mientras que otras son útiles de manera más general. Complementan la interfaz programática Scheme de Guix de modo conveniente.

7.1 Invocación de guix build

La orden `guix build` construye paquetes o derivaciones y sus dependencias, e imprime las rutas del almacen resultantes. Fíjese que no modifica el perfil de la usuaria—este es el trabajo de la orden `guix package` (véase Sección 4.2 [Invocación de guix package], página 36). Por tanto, es útil principalmente para las desarrolladoras de la distribución.

La sintaxis general es:

```
guix build opciones paquete-o-derivación...
```

Como ejemplo, la siguiente orden construye las últimas versiones de Emacs y Guile, muestra sus log de construcción, y finalmente muestra los directorios resultantes:

```
guix build emacs guile
```

De forma similar, la siguiente orden construye todos los paquetes disponibles:

```
guix build --quiet --keep-going \ 
  'guix package -A | cut -f1,2 --output-delimiter=@'
```

`paquete-o-derivación` puede ser tanto el nombre de un paquete que se encuentra en la distribución de software como `coreutils` o `coreutils@8.20`, o una derivación como `/gnu/store/...-coreutils-8.19.drv`. En el primer caso, el paquete de nombre (y opcionalmente versión) correspondiente se busca entre los módulos de la distribución GNU (véase Sección 6.1 [Módulos de paquetes], página 76).

De manera alternativa, la opción `--expression` puede ser usada para especificar una expresión Scheme que evalúa a un paquete; esto es útil para diferenciar entre varios paquetes con el mismo nombre o si se necesitan variaciones del paquete.

Puede haber cero o más `opciones`. Las opciones disponibles se describen en la subsección siguiente.

7.1.1 Opciones comunes de construcción

Un número de opciones que controlan el proceso de construcción son comunes a `guix build` y otras órdenes que pueden lanzar construcciones, como `guix package` o `guix archive`. Son las siguientes:

```
--load-path=directorio
-L directorio
```

Añade `directorio` al frente de la ruta de búsqueda de módulos de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76).

Esto permite a las usuarias definir sus propios paquetes y hacerlos visibles a las herramientas de línea de órdenes.

```
--keep-failed
-K
```

Mantiene los árboles de construcción de las construcciones fallidas. Por tanto, si una construcción falla, su árbol de construcción se mantiene bajo `/tmp`, en
un directorio cuyo nombre se muestra al final del log de construcción. Esto es útil cuando se depuran problemas en la construcción. Véase Sección 7.1.4 [Depuración de fallos de construcción], página 127, para trucos y consejos sobre cómo depurar problemas en la construcción.

Esta opción implica \texttt{--no-offload}, y no tiene efecto cuando se conecta a un daemon remoto con una URI \texttt{guix://} (véase Sección 6.4 \[El almacén], página 98).

\texttt{--keep-going} \texttt{-k}  
Seguir adelante cuando alguna de las derivaciones de un fallo durante la construcción; devuelve una única vez todas las construcciones que se han completado o bien han fallado.

El comportamiento predeterminado es parar tan pronto una de las derivaciones especificadas falle.

\texttt{--dry-run} \texttt{-n}  
No construye las derivaciones.

\texttt{--fallback}  
Cuando la sustitución de un binario preconstruido falle, intenta la construcción local de paquetes (véase Sección 4.3.5 \[Fallos en las sustituciones\], página 46).

\texttt{--substitute-urls=urls}  
Considera \texttt{urls} la lista separada por espacios de URLs de fuentes de sustituciones, anulando la lista predeterminada de URLs de \texttt{guix-daemon} (véase \[guix-daemon URLs\], página 16).

Significa que las sustituciones puede ser descargadas de \texttt{urls}, mientras que estén firmadas por una clave autorizada por la administradora del sistema (véase Sección 4.3 \[Sustituciones\], página 44).

Cuando \texttt{urls} es la cadena vacía, las sustituciones están efectivamente desactivadas.

\texttt{--no-substitutes}  
No usa sustituciones para la construcción de productos. Esto es, siempre realiza las construcciones localmente en vez de permitir la descarga de binarios preconstruidos (véase Sección 4.3 \[Sustituciones\], página 44).

\texttt{--no-grafts}  
No “injerta” paquetes. En la práctica esto significa que las actualizaciones de paquetes disponibles como injertos no se aplican. Véase Capítulo 11 \[Actualizaciones de seguridad\], página 425, para más información sobre los injertos.

\texttt{--rounds=n}  
Construye cada derivación \texttt{n} veces seguidas, y lanza un error si los resultados de las construcciones consecutivas no son idénticos bit-a-bit.

Esto es útil para la detección de procesos de construcción no-deterministas. Los procesos de construcción no-deterministas son un problema puesto que prácticamente imposibilitan a las usuarias la \textit{verificación} de la autenticidad de binarios proporcionados por terceras partes. Véase Sección 7.11 \[Invocación de guix challenge\], página 153, para más sobre esto.
Fíjese que, actualmente, los resultados de las construcciones discordantes no se mantienen, por lo que debe que investigar manualmente en caso de un error—por ejemplo, mediante la extracción de uno de los resultados con guix archive --export (véase Sección 4.11 [Invocación de guix archive], página 63), seguida de una reconstrucción, y finalmente la comparación de los dos resultados.

--no-offload
No usa la delegación de construcciones en otras máquinas (véase Sección 2.4.2 [Configuración de delegación del daemon], página 11). Es decir, siempre realiza las construcciones de manera local en vez de delegar construcciones a máquinas remotas.

--max-silent-time=segundos
Cuando la construcción o sustitución permanece en silencio más de segundos, la finaliza e informa de un fallo de construcción.

Por defecto, se respeta la configuración del daemon (véase Sección 2.5 [Invocación de guix-daemon], página 15).

--timeout=segundos
Del mismo modo, cuando el proceso de construcción o sustitución dura más de segundos, lo termina e informa un fallo de construcción.

Por defecto, se respeta la configuración del daemon (véase Sección 2.5 [Invocación de guix-daemon], página 15).

-v nivel
--verbosity=nivel
Usa el nivel de detalle especificado, un entero. Seleccionar 0 significa que no se produce ninguna salida, 1 es para salida silenciosa y 2 muestra toda la salida del registro de construcción en la salida estándar de error.

--cores=n
-c n
Permite usar n núcleos de la CPU para la construcción. El valor especial 0 significa usar tantos como núcleos haya en la CPU.

--max-jobs=n
-M n
Permite como máximo n trabajos de construcción en paralelo. Véase Sección 2.5 [Invocación de guix-daemon], página 15, para detalles acerca de esta opción y la opción equivalente de guix-daemon.

--debug=nivel
Usa el nivel de detalle proporcionado en los mensajes procedentes del daemon de construcción. nivel debe ser un entero entre 0 y 5; valores mayores indican una salida más detallada. Establecer un nivel de 4 o superior puede ser útil en la depuración de problemas de configuración con el daemon de construcción.

Tras las cortinas, guix build es esencialmente una interfaz al procedimiento package-derivation del módulo (guix packages), y al procedimiento build-derivations del módulo (guix derivations).

Además de las opciones proporcionadas explícitamente en la línea de órdenes, guix build y otras órdenes guix que permiten la construcción respetan el contenido de la variable de entorno GUIX_BUILD_OPTIONS.
Las usuarias pueden definir esta variable para que contenga una lista de opciones de línea de órdenes que se usarán automáticamente por `guix build` y otras órdenes `guix` que puedan realizar construcciones, como en el ejemplo siguiente:

```
$ export GUIX_BUILD_OPTIONS="--no-substitutes -c 2 -L /foo/bar"
```

Estas opciones se analizan independientemente, y el resultado se añade a continuación de las opciones de línea de órdenes.

### 7.1.2 Opciones de transformación de paquetes

Otro conjunto de opciones de línea de órdenes permitidas por `guix build` y también `guix package` son las opciones de transformación de paquetes. Son opciones que hacen posible la definición de variaciones de paquetes—por ejemplo, paquetes construidos con un código fuente diferente. Es una forma conveniente de crear paquetes personalizados al vuelo sin tener que escribir las definiciones de las variaciones del paquete (véase Sección 6.2 [Definición de paquetes], página 77).

```
--with-source=fuente
--with-source=paquete=fuente
--with-source=paquete@versión=fuente
```

Usa `fuente` como la fuente de `paquete`, y versión como su número de versión. `fuente` debe ser un nombre de fichero o una URL, como en `guix download` (véase Sección 7.3 [Invocación de guix download], página 129).

Cuando se omite `paquete`, se toma el nombre de paquete especificado en la línea de órdenes que coincida con el nombre base de `fuente`—por ejemplo, si `fuente` fuese `/src/guile-2.0.10.tar.gz`, el paquete correspondiente sería `guile`.

Del mismo modo, si se omite versión, la cadena de versión se deduce de `fuente`; en el ejemplo previo sería `2.0.10`.

Esta opción permite a las usuarias probar versiones del paquete distintas a las proporcionadas en la distribución. El ejemplo siguiente descarga `ed-1.7.tar.gz` de un espejo GNU y lo usa como la fuente para el paquete `ed`:

```
guix build ed --with-source=mirror://gnu/ed/ed-1.7.tar.gz
```

Como desarrolladora, `--with-source` facilita la prueba de versiones candidatas para la publicación:

```
guix build guile --with-source=../guile-2.0.9.219-e1bb7.tar.xz
```

... o la construcción desde una revisión en un entorno limpio:

```
$ git clone git://git.sv.gnu.org/guix.git
$ guix build guix --with-source=guix@1.0=.
```

```
--with-input=paquete=reemplazo
```

Substituye dependencias de `paquete` por dependencias de `reemplazo`. `paquete` debe ser un nombre de paquete, y `reemplazo` debe ser una especificación de paquete como `guile` o `guile@1.8`.

Por ejemplo, la orden siguiente construye Guix, pero substituye su dependencia de la versión estable actual de Guile con una dependencia en la versión antigua de Guile, `guile@2.0`:

```
guix build --with-input=guile=guile@2.0 guix
```
Esta sustitución se realiza de forma recursiva y en profundidad. Por lo que en este ejemplo, tanto guix como su dependencia guile-json (que también depende de guile) se reconstruyen contra guile@2.0.

Se implementa usando el procedimiento Scheme package-input-rewriting (véase Sección 6.2 [Definición de paquetes], página 77).

**--with-graft=paquete=reemplazo**
Es similar a **--with-input** pero con una diferencia importante: en vez de reconstruir la cadena de dependencias completa, *reemplazo* se construye y se *injerta* en los binarios que inicialmente hacían referencia a *paquete*. Véase Capítulo 11 [Actualizaciones de seguridad], página 425, para más información sobre injertos.

Por ejemplo, la orden siguiente injerta la versión 3.5.4 de GnuTLS en Wget y todas sus dependencias, substituyendo las referencias a la versión de GnuTLS que tienen actualmente:

```
guix build --with-graft=gnutls=gnutls@3.5.4 wget
```

Esta opción tiene la ventaja de ser mucho más rápida que la reconstrucción de todo. Pero hay una trampa: funciona si y solo si *paquete* y *reemplazo* son estrictamente compatibles—por ejemplo, si proporcionan una biblioteca, la interfaz binaria de aplicación (ABI) de dichas bibliotecas debe ser compatible. Si *reemplazo* es incompatible de alguna manera con *paquete*, el paquete resultante puede no ser usable. ¡Usela con precaución!

**--with-git-url=paquete=url**
Construye *paquete* desde la última revisión de la rama *rama* del repositorio Git en *url*. Los submódulos del repositorio Git se obtienen de forma recursiva.

Por ejemplo, la siguiente orden construye la biblioteca NumPy de Python contra la última revisión de la rama master de Python en sí:

```
guix build python-numpy \ 
   --with-git-url=python=https://github.com/python/cpython
```

Esta opción también puede combinarse con **--with-branch** o **--with-commit** (véase más adelante).

Obviamente, ya que se usa la última revisión de la rama proporcionada, el resultado de dicha orden varía con el tiempo. No obstante es una forma conveniente de reconstruir una pila completa de software contra las últimas revisiones de uno o varios paquetes. Esto es particularmente útil en el contexto de integración continua (CI).

Los directorios de trabajo se conservan en caché en `~/.cache/guix/checkouts` para agilizar accesos consecutivos al mismo repositorio. Puede desear limpiarla de vez en cuando para ahorrar espacio en el disco.

**--with-branch=paquete=rama**
Construye *paquete* desde la última revisión de *rama*. Si el campo *source* de *paquete* es un origen con el método *git-fetch* (véase Sección 6.2.2 [Referencia de origen], página 83) o un objeto *git-checkout*, la URL del repositorio se toma de dicho campo *source*. En otro caso, se debe especificar la URL del repositorio Git mediante el uso de **--with-git-url**.
Por ejemplo, la siguiente orden construye `guile-sqlite3` desde la última revisión de su rama `master` y, una vez hecho, construye `guix` (que depende de él) y `cuirass` (que depende de `guix`) en base a esta construcción específica de `guile-sqlite3`:

```bash
guix build --with-branch=guile-sqlite3=master cuirass
```

```
--with-commit=paquete=revisión
```

Esta opción es similar a `--with-branch`, salvo que construye desde `revisión` en vez de usar la punta de la rama. `revisión` debe ser un identificador SHA-1 de revisión Git válido o una etiqueta.

### 7.1.3 Opciones de construcción adicionales

Las opciones de línea de ordenes presentadas a continuación son específicas de `guix build`.

```
--quiet
-q
```

Construye silenciosamente, sin mostrar el registro de construcción; es equivalente a `--verbosity=0`. Al finalizar, el registro de construcción se mantiene en `/var` (o similar) y puede recuperarse siempre mediante el uso de la opción `--log-file`.

```
--file=fichero
-f fichero
```

Construye el paquete, derivación u otro objeto tipo-fichero al que evalúa el código en `fichero` (véase Sección 6.7 [Expresiones-G], página 108). Como un ejemplo, `fichero` puede contener una definición como esta (véase Sección 6.2 [Definición de paquetes], página 77):

```scheme
(use-modules (guix)
  (guix build-system gnu)
  (guix licenses))

(package
  (name "hello")
  (version "2.10")
  (source (origin
    (method url-fetch)
    (uri (string-append "mirror://gnu/hello/hello-" version ".tar.gz"))
    (sha256
      (base32
        "0ssi1wpaf7plaswqjwigppsg5fyh99vdlb9kz17c91ng89ndq1i")))
    (build-system gnu-build-system)
  (synopsis "Hello, GNU world: An example GNU package")
  (description "Guess what GNU Hello prints!")
  (home-page "http://www.gnu.org/software/hello/")
  (license gpl3+))
```
Capítulo 7: Utilidades

--manifest=manifest
-m manifest

Build all packages listed in the given manifest (véase [profile-manifest], página 39).

--expression=expr
-e expr

Construye el paquete o derivación a la que evalúa expr.

Por ejemplo, expr puede ser (@ (gnu packages guile) guile-1.8), que designa sin ambigüedad a esta variante específica de la versión 1.8 de Guile.

De manera alternativa, expr puede ser una expresión-G, en cuyo caso se usa como un programa de construcción pasado a gexp->derivation (véase Sección 6.7 [Expresiones-G], página 108).

Por último, expr puede hacer referencia a un procedimiento monádico sin parámetros (véase Sección 6.6 [La monáda del almacén], página 104). El procedimiento debe devolver una derivación como un valor monádico, el cual después se pasa a través de run-with-store.

--source
-S

Construye las derivaciones de las fuentes de los paquetes, en vez de los paquetes mismos.

Por ejemplo, guix build -S gcc devuelve algo como /gnu/store/...-gcc-4.7.2.tar.bz2, el cual es el archivador tar de fuentes de GCC.

El archivador tar devuelto es el resultado de aplicar cualquier parche y fragmento de código en el origen (campo origin) del paquete (véase Sección 6.2 [Definición de paquetes], página 77).

Tenga en cuenta que guix build -S compila las fuentes únicamente de los paquetes especificados. Ésto no incluye las dependencias enlazadas estáticamente y por sí mismas son insuficientes para reproducir los paquetes.

--sources

Obtiene y devuelve las fuentes de paquete-o-derivación y todas sus dependencias, recursivamente. Ésto es útil para obtener una copia local de todo el código fuente necesario para construir los paquetes, le permite construirlos cuando el momento sin acceso a la red. Es una extensión de la opción --source y puede aceptar uno de los siguientes valores opcionales como parámetro:

package Este valor hace que la opción --sources se comporte de la misma manera que la opción --source.

all Construye las derivaciones de las fuentes de todos los paquetes, incluyendo cualquier fuente que pueda enumerarse como entrada (campo inputs). Éste es el valor predeterminado.

$ guix build --sources tzdata
The following derivations will be built:
 /gnu/store/...-tzdata2015b.tar.gz.drv
 /gnu/store/...-tzcode2015b.tar.gz.drv

transitive

Construye las derivaciones de fuentes de todos los paquetes, así como todas las entradas transitivas de los paquetes. Ésto puede
usarse, por ejemplo, para obtener las fuentes de paquetes para una construcción posterior sin conexión a la red.

```
$ guix build --sources=transitive tzdata
The following derivations will be built:
 /gnu/store/...-tzcode2015b.tar.gz.drv
 /gnu/store/...-findutils-4.4.2.tar.xz.drv
 /gnu/store/...-grep-2.21.tar.xz.drv
 /gnu/store/...-coreutils-8.23.tar.xz.drv
 /gnu/store/...-make-4.1.tar.xz.drv
 /gnu/store/...-bash-4.3.tar.xz.drv

... 
```

```
--system=sistema
-s sistema
```

Intenta la construcción para sistema—por ejemplo, i686-linux—en vez del tipo de sistema de la máquina de construcción. La orden guix build le permite repetir esta opción varias veces, en cuyo caso construye para todos los sistemas especificados; otras ordenes ignoran opciones -s extrañas.

**Nota:** La opción --system es para compilación nativa y no debe confundirse con la compilación cruzada. Véase --target más adelante para información sobre compilación cruzada.

Un ejemplo de uso de esta opción es en sistemas basados en Linux, que pueden emular diferentes personalidades. Por ejemplo, pasar --system=i686-linux en un sistema x86_64-linux o --system=armhf-linux en un sistema aarch64-linux le permite construir paquetes en un entorno de 32-bits completo.

**Nota:** La construcción para un sistema armhf-linux está disponible de manera incondicional en máquinas aarch64-linux, aunque determinados procesadores aarch64 no lo permitan, notablemente el ThunderX.

De manera similar, cuando la emulación transparente con QEMU y binfmt_misc está activada (véase Sección 8.8.24 [Servicios de virtualización], página 353), puede construir para cualquier sistema para el que un manejador QEMU de binfmt_misc esté instalado.

Las construcciones para un sistema distinto al de la máquina que usa se pueden delegar también a una máquina remota de la arquitectura correcta. Véase Sección 2.4.2 [Configuración de delegación del daemon], página 11, para más información sobre delegación.

```
--target=tripleta
```

Compilación cruzada para la tripleta, que debe ser una tripleta GNU válida, como "mips64el-linux-gnu" (véase Sección “Specifying Target triplets” en Autoconf).

```
--check
```

Reconstruye paquete-o-derivación, que ya está disponible en el almacén, y emite un error si los resultados de la construcción no son idénticos bit-a-bit.

Este mecanismo le permite comprobar si sustituciones previamente instaladas son genuinas (véase Sección 4.3 [Sustituciones], página 44), o si el resultado de la
construcción de un paquete es determinista. Véase Sección 7.11 [Invocación de guix challenge], página 153, para más información de referencia y herramientas. Cuando se usa conjuntamente con --keep-failed, la salida que difiere se mantiene en el almacén, bajo /gnu/store/...-check. Esto hace fácil buscar diferencias entre los dos resultados.

--repair Intenta reparar los elementos del almacén especificados, si están corruptos, volviendo a descargarlos o mediante su reconstrucción. Esta operación no es atómica y por lo tanto está restringida a root.

--derivations
-d Devuelve las rutas de derivación, no las rutas de salida, de los paquetes proporcionados.

--root=fichero
-r fichero Hace que fichero sea un enlace simbólico al resultado, y lo registra como una raíz del recolector de basura. Consecuentemente, los resultados de esta invocación de guix build se protegen de la recolección de basura hasta que fichero se elimine. Cuando se omite esa opción, los resultados son candidatos a la recolección de basura en cuanto la construcción se haya completado. Véase Sección 4.5 [Invocación de guix gc], página 48, para más sobre las raíces del recolector de basura.

--log-file Devuelve los nombres de ficheros o URL de los log de construcción para el paquete-o-derivación proporcionado, o emite un error si no se encuentran los log de construcción. Esto funciona independientemente de cómo se especificasen los paquetes o derivaciones. Por ejemplo, las siguientes invocaciones son equivalentes:

```
guix build --log-file 'guix build -d guile'
guix build --log-file 'guix build guile'
guix build --log-file guile
```

Si un log no está disponible localmente, y a menos que se proporcione --no-substitutes, la orden busca el log correspondiente en uno de los servidores de sustituciones (como se especificaron con --substitute-urls).

Por lo que dado el caso, imagínese que desea ver el log de construcción de GDB en MIPS, pero realmente está en una máquina x86_64:

```
$ guix build --log-file gdb -s mips64el-linux
https://ci.guix.gnu.org/log/...gdb-7.10
```

¡Puede acceder libremente a una biblioteca inmensa de log de construcción!

7.1.4 Depuración de fallos de construcción

Cuando esté definiendo un paquete nuevo (véase Sección 6.2 [Definición de paquetes], página 77), probablemente se encuentre que dedicando algún tiempo a depurar y afinar
la construcción hasta obtener un resultado satisfactorio. Para hacerlo, tiene que lanzar manualmente las órdenes de construcción en un entorno tan similar como sea posible al que el daemon de construcción usa.

Para ello, la primera cosa a hacer es usar la opción `--keep-failed` o `-K` de `guix build`, lo que mantiene el árbol de la construcción fallida en `/tmp` o el directorio que especifique con `TMPDIR` (véase Sección 7.1 [Invocación de guix build], página 119).

De ahí en adelante, puede usar `cd` para ir al árbol de la construcción fallida y cargar el fichero `environment-variables`, que contiene todas las definiciones de variables de entorno que existían cuando la construcción falló. Digamos que está depurando un fallo en la construcción del paquete `foo`; una sesión típica sería así:

```
$ guix build foo -K
... build fails
$ cd /tmp/guix-build-foo.drv-0
$ source ./environment-variables
$ cd foo-1.2
```

Ahora puede invocar órdenes (casi) como si fuese el daemon y encontrar los errores en su proceso de construcción.

A veces ocurre que, por ejemplo, las pruebas de un paquete pasan cuando las ejecuta manualmente pero fallan cuando el daemon las ejecuta. Esto puede suceder debido a que el daemon construye dentro de contenedores donde, al contrario que en nuestro entorno previo, el acceso a la red no está disponible, `/bin/sh` no existe, etc. (véase Sección 2.4.1 [Configuración del entorno de construcción], página 9).

En esos casos, puede tener que inspeccionar el proceso de construcción desde un contenedor similar al creado por el daemon de construcción:

```
$ guix build -K foo
...
$ cd /tmp/guix-build-foo.drv-0
$ guix environment --no-grafts -C foo --ad-hoc strace gdb
[env]# source ./environment-variables
[env]# cd foo-1.2
```

Aquí, `guix environment -C` crea un contenedor y lanza un shell nuevo en él (véase Sección 5.1 [Invocación de guix environment], página 66). El fragmento `--ad-hoc strace gdb` añade las órdenes `strace` y `gdb` al contenedor, las cuales pueden resultar útiles durante la depuración. La opción `--no-grafts` asegura que obtenemos exactamente el mismo entorno, con paquetes sin injertos (véase Capítulo 11 [Actualizaciones de seguridad], página 425, para más información sobre los injertos).

Para acercarnos más al contenedor usado por el daemon de construcción, podemos eliminar `/bin/sh`:

```
[env]# rm /bin/sh
```

(No se preocupe, es inocuo: todo esto ocurre en el contenedor de usar y tirar creado por `guix environment`).

La orden `strace` probablemente no esté en la ruta de búsqueda, pero podemos ejecutar:

```
[env]# $GUIX_ENVIRONMENT/bin/strace -f -o log make check
```
De este modo, no solo habrá reproducido las variables de entorno que usa el daemon, también estará ejecutando el proceso de construcción en un contenedor similar al usado por el daemon.

### 7.2 Invocación de guix edit

¡Tantos paquetes, tantos ficheros de fuentes! La orden `guix edit` facilita la vida de las usuarias y empaquetadoras apuntando su editor al fichero de fuentes que contiene la definición de los paquetes especificados. Por ejemplo:

```
guix edit gcc@4.9 vim
```

lanza el programa especificado en la variable de entorno `VISUAL` o en `EDITOR` para ver la receta de GCC 4.9.3 y la de Vim.

Si está usando una copia de trabajo de Git de Guix (véase Sección 14.1 [Construcción desde Git], página 435), o ha creado sus propios paquetes en `GUIX_PACKAGE_PATH` (véase Sección 6.1 [Módulos de paquetes], página 76), será capaz de editar las recetas de los paquetes. En otros casos, podrá examinar las recetas en modo de lectura únicamente para paquetes actualmente en el almacén.

En vez de `GUIX_PACKAGE_PATH`, la opción de línea de órdenes `--load-path=directorio` (o en versión corta `--L directorio`) le permite añadir `directorio` al inicio de la ruta de búsqueda de módulos de paquete y por tanto hacer sus propios paquetes visibles.

### 7.3 Invocación de guix download

Durante la escritura de una definición de paquete, las desarrolladoras típicamente tienen que descargar un archivo tar de fuentes, calcular su hash SHA256 y escribir ese hash en la definición del paquete (véase Sección 6.2 [Definición de paquetes], página 77). La herramienta `guix download` ayuda con esta tarea: descarga un fichero de la URI proporcionada, lo añade al almacén e imprime tanto su nombre de fichero en el almacén como su hash SHA256.

El hecho de que el fichero descargado se añada al almacén ahorra ancho de banda: cuando el desarrollador intenta construir el paquete recién definido con `guix build`, el archivador de fuentes no tiene que descargarlo de nuevo porque ya está en el almacén. También es una forma conveniente de conservar ficheros temporalmente, que pueden ser borrados en un momento dado (véase Sección 4.5 [Invocación de guix gc], página 48).

La orden `guix download` acepta las mismas URI que las usadas en las definiciones de paquetes. En particular, permite URI `mirror://`. Las URI `https` (HTTP sobre TLS) se aceptan cuando el enlace Guile con GnuTLS está disponible en el entorno de la usuaria; cuando no está disponible se emite un error. Véase Sección “Guile Preparations” en `GnuTLS-Guile`, para más información.

`guix download` verifica los certificados del servidor HTTPS cargando las autoridades X.509 del directorio al que apunta la variable de entorno `SSL_CERT_DIR` (véase Sección 8.10 [Certificados X.509], página 390), a menos que se use `--no-check-certificate`.

Las siguientes opciones están disponibles:
--format=fmt
-f fmt Escribe el hash en el formato especificado por fmt. Para más información sobre los valores aceptados en fmt, véase Sección 7.4 [Invocación de guix hash], página 130.

--no-check-certificate
No valida los certificados X.509 de los servidores HTTPS. Cuando se usa esta opción, no tiene absolutamente ninguna garantía de que está comunicando con el servidor responsable de la URL auténtico, lo que le hace vulnerables a ataques de interceptación (“man-in-the-middle”).

--output=fichero
-o fichero Almacena el fichero descargado en fichero en vez de añadirlo al almacén.

7.4 Invocación de guix hash

La orden guix hash calcula el hash SHA256 de un fichero. Es principalmente una conveniente herramienta para cualquiera que contribuya a la distribución: calcula el hash criptográfico de un fichero, que puede usarse en la definición de un paquete (véase Sección 6.2 [Definición de paquetes], página 77).

La sintaxis general es:

    guix hash opciones fichero

Cuando fichero es - (un guión), guix hash calcula el hash de los datos leídos por la entrada estándar. guix hash tiene las siguientes opciones:

--format=fmt
-f fmt Escribe el hash en el formato especificado por fmt.

Los formatos disponibles son: nix-base32, base32, base16 (se puede usar también hex y hexadecimal).

Si no se especifica la opción --format, guix hash mostrará el hash en nix-base32. Esta representación es la usada en las definiciones de paquetes.

--recursive
-r Calcula el hash de fichero recursivamente.

Es este caso el hash se calcula en un archivador que contiene fichero, incluyendo su contenido si es un directorio. Algunos de los metadatos de fichero son parte del archivador; por ejemplo, cuando fichero es un fichero normal, el hash es diferente dependiendo de si fichero es ejecutable o no. Los metadatos como las marcas de tiempo no influyen en el hash (véase Sección 4.11 [Invocación de guix archive], página 63).

--exclude-vcs
-x Cuando se combina con --recursive, excluye los directorios del sistema de control de versiones (.bzr, .git, .hg, etc.).

Como un ejemplo, así es como calcularía el hash de una copia de trabajo Git, lo cual es útil cuando se usa el método git-fetch (véase Sección 6.2.2 [Referencia de origen], página 83):

    $ git clone http://example.org/foo.git
Capítulo 7: Utilidades

7.5 Invocación de guix import

La orden `guix import` es útil para quienes desean añadir un paquete a la distribución con el menor trabajo posible—una demanda legítima. La orden conoce algunos repositorios de los que puede “importar” metadatos de paquetes. El resultado es una definición de paquete, o una plantilla de ella, en el formato que conocemos (véase Sección 6.2 [Definición de paquetes], página 77).

La sintaxis general es:

```
$ guix import importador opciones...
```

`importador` especifica la fuente de la que se importan los metadatos del paquete, `opciones` especifica un identificador de paquete y otras opciones específicas del `importador`.

Algunos de los importadores dependen de poder ejecutar la orden `gpgv`. Para ello, GnuPG debe estar instalado y en `$PATH`; ejecute `guix install gnupg` si es necesario.

Actualmente los “importadores” disponibles son:

**gnu**

Importa los metadatos del paquete GNU seleccionado. Proporciona una plantilla para la última versión de dicho paquete GNU, incluyendo el hash de su archivador tar de fuentes, y su sinopsis y descripción canónica.

Información adicional como las dependencias del paquete y su licencia deben ser deducidas manualmente.

Por ejemplo, la siguiente orden devuelve una definición de paquete para GNU Hello.

```
$ guix import gnu hello
```

Las opciones específicas de línea de ordenes son:

```
--key-download=política
```

Como en `guix refresh`, especifica la política de tratamiento de las claves OpenPGP no encontradas cuando se verifica la firma del paquete. Véase Sección 7.6 [Invocación de guix refresh], página 136.

**pypi**

Importa metadatos desde el índice de paquetes Python (PyPI) (https://pypi.python.org/). La información se toma de la descripción con formato JSON disponible en pypi.python.org y habitualmente incluye toda la información relevante, incluyendo las dependencias del paquete. Para una máxima eficiencia, se recomienda la instalación de la utilidad `unzip`, de manera que el importador pueda extraer los archivos wheel de Python y obtener datos de ellos.

La siguiente orden importa los metadatos para el paquete de Python `itsdangerous`:

```
$ guix import pypi itsdangerous
```

```
--recursive
-r
```

Recorre el grafo de dependencias del paquete original proporcionado recursivamente y genera expresiones de paquete para todos aquellos paquetes que no estén todavía en Guix.
**gem** Importa metadatos desde RubyGems ([https://rubygems.org/](https://rubygems.org/)). La información se extrae de la descripción en formato JSON disponible en [rubygems.org](https://rubygems.org) e incluye la información más relevante, incluyendo las dependencias en tiempo de ejecución. Hay algunos puntos a tener en cuenta, no obstante. Los metadatos no distinguen entre sinopsis y descripción, por lo que se usa la misma cadena para ambos campos. Adicionalmente, los detalles de las dependencias no-Ruby necesarias para construir extensiones nativas no está disponible y se deja como ejercicio a la empaquetadora.

La siguiente orden importa los meta-datos para el paquete de Ruby `rails`:

```
guix import gem rails --recursive
```

Recorre el grafo de dependencias del paquete original proporcionado recursivamente y genera expresiones de paquete para todos aquellos paquetes que no estén todavía en Guix.

**cpan** Importa metadatos desde MetaCPAN ([https://www.metacpan.org/](https://www.metacpan.org/)). La información se extrae de la descripción en formato JSON disponible a través del API de MetaCPAN ([https://fastapi.metacpan.org/](https://fastapi.metacpan.org/)) e incluye la información más relevante, como las dependencias de otros módulos. La información de la licencia debe ser comprobada atentamente. Si Perl está disponible en el almacén, se usará la utilidad `corelist` para borrar los módulos básicos de la lista de dependencias.

La siguiente orden importa los metadatos del módulo Perl `Acme::Boolean`:

```
guix import cpan Acme::Boolean
```

**cran** Importa metadatos desde CRAN ([https://cran.r-project.org/](https://cran.r-project.org/)), el repositorio central para el entorno estadístico y gráfico GNU R ([https://r-project.org](https://r-project.org)).

La información se extrae del fichero `DESCRIPTION` del paquete.

La siguiente orden importa los metadatos del paquete de R `Cairo`:

```
guix import cran Cairo
```

Cuando se añade `--recursive`, el importador recorrerá el grafo de dependencias del paquete original proporcionado recursivamente y generará expresiones de paquetes para todos aquellos que no estén todavía en Guix.

Cuando se agrega `--archive=bioconductor`, los metadatos se importan de Bioconductor ([https://www.bioconductor.org](https://www.bioconductor.org)), un repositorio de paquetes R para el análisis y comprensión de datos genéticos de alto caudal en bioinformática.

La información se extrae del fichero `DESCRIPTION` que contiene el archivo del paquete.

La siguiente orden importa los metadatos del paquete de R `GenomicRanges`:

```
guix import cran --archive=bioconductor GenomicRanges
```

Por último, también puede importar paquetes de R que no se hayan publicado todavía en CRAN o en Bioconductor siempre que estén en un repositorio git. Use `--archive=git` seguido de la URL del repositorio git:
guix import cran --archive=git https://github.com/immunogenomics/harmony

**texlive**

Importa metadatos desde CTAN (https://www.ctan.org/), la completa red de archivos TeX para paquetes TeX que son parte de la distribución TeX Live (https://www.tug.org/texlive/).

La información del paquete se obtiene a través del API XML proporcionado por CTAN, mientras que el código fuente se descarga del repositorio SVN del proyecto TeX Live. Se hace porque CTAN no guarda archivos con versiones.

La siguiente orden importa los metadatos del paquete de TeX **fontspec**:

```
guix import texlive fontspec
```

Cuando se añade --archive=DIRECTORIO, el código fuente no se descarga del subdirectorio latex del árbol texmf-dist/source en el repositorio SVN de Tex Live, sino de el directorio especificado bajo la misma raíz.

La siguiente orden importa los metadatos del paquete **ifxetex** de CTAN mientras que obtiene las fuentes del directorio texmf/source/generic:

```
guix import texlive --archive=generic ifxetex
```

**json**

Importa metadatos de paquetes desde un fichero JSON local. Considere el siguiente ejemplo de definición de paquete en formato JSON:

```
{
  "name": "hello",
  "version": "2.10",
  "source": "mirror://gnu/hello/hello-2.10.tar.gz",
  "build-system": "gnu",
  "home-page": "https://www.gnu.org/software/hello/",
  "synopsis": "Hello, GNU world: An example GNU package",
  "description": "GNU Hello prints a greeting.",
  "license": "GPL-3.0+",
  "native-inputs": ["gettext"]
}
```

Los nombres de los campos son los mismos que para el registro <package> (Véase Sección 6.2 [Definición de paquetes], página 77). Las referencias a otros paquetes se proporcionan como listas JSON de cadenas de especificación de paquete entrecomilladas como guile o guile@2.0.

El importador también permite una definición de fuentes más explícita usando los campos comunes de los registros <origin>:

```
{
  ...
  "source": {
    "method": "url-fetch",
    "uri": "mirror://gnu/hello/hello-2.10.tar.gz",
    "sha256": {
      "base32": "0ssi1wpaf7plawqqjwigppsg5fyyh99vdlb9zkzl7c91ng89ndq1i"
    }
  }
  ...
```
La siguiente orden importa los metadatos desde el fichero JSON `hello.json` y devuelve una expresión de “package”:

```
guix import json hello.json
```

**nix** Importa metadatos desde una copia local de las fuentes de la distribución Nixpkgs ([https://nixos.org/nixpkgs/](https://nixos.org/nixpkgs/)). Las definiciones de paquete en Nixpkgs típicamente están escritas en una mezcla de lenguaje Nix y código Bash. Esta orden únicamente importa la estructura de alto nivel del paquete escrita en lenguaje Nix. Normalmente incluye todos los campos básicos de una definición de paquete.

Cuando se importa un paquete GNU, la sinopsis y la descripción se substituyen por la variante canónica oficial.

Habitualmente, tendrá que ejecutar primero:

```
export NIX_REMOTE=daemon
```
de modo que `nix-instantiate` no intente abrir la base de datos Nix.

Como un ejemplo, la orden siguiente importa la definición de paquete de LibreOffice (más precisamente, importa la definición del paquete asociado al atributo de nivel superior `libreoffice`):

```
guix import nix ~/path/to/nixpkgs libreoffice
```

**hackage** Importa metadatos desde el archivo central de paquetes de la comunidad Haskell Hackage ([https://hackage.haskell.org/](https://hackage.haskell.org/)). La información se obtiene de ficheros Cabal e incluye toda la información relevante, incluyendo las dependencias del paquete.

Las opciones específicas de línea de ordenes son:

- `--stdin`
  - `s` Lee un fichero Cabal por la entrada estándar.

- `--no-test-dependencies`
  - `t` No incluye las dependencias necesarias únicamente para las baterías de pruebas.

- `--cabal-environment=alist`
  - `e alist` *alist* es una lista asociativa Scheme que define el entorno en el que los condicionales Cabal se evalúan. Los valores aceptados son: `os`, `arch`, `impl` y una cadena que representa el nombre de la condición. El valor asociado a la condición tiene que ser o bien el símbolo `true` o bien `false`. Los valores predeterminados asociados a las claves `os`, `arch` y `impl` son ‘`linux`’, ‘`x86_64`’ y ‘`ghc`’, respectivamente.

- `--recursive`
  - `r` Recorre el grafo de dependencias del paquete original proporcionado recursivamente y genera expresiones de paquete para todos aquellos paquetes que no estén todavía en Guix.

---

1. Esto depende de la orden `nix-instantiate` de Nix ([https://nixos.org/nix/](https://nixos.org/nix/)).
La siguiente orden importa los metadatos de la última versión del paquete Haskell HTTP sin incluir las dependencias de las pruebas y especificando la opción ‘network-uri’ con valor false:

```
guix import hackage -t -e "'(("network-uri" . false))" HTTP
```

Se puede especificar opcionalmente una versión específica del paquete añadiendo al nombre del paquete una arroba y el número de versión como en el siguiente ejemplo:

```
guix import hackage mtl@2.1.3.1
```

**stackage**

El importador stackage es un recubrimiento sobre el de hackage. Toma un nombre de paquete, busca la versión de paquete incluida en una publicación de la versión de mantenimiento extendido (LTS) Stackage (https://www.stackage.org) y usa el importador hackage para obtener sus metadatos. Fíjese que es su decisión seleccionar una publicación LTS compatible con el compilador GHC usado en Guix.

Las opciones específicas de línea de ordenes son:

--no-test-dependencies

```
t
```

No incluye las dependencias necesarias únicamente para las baterías de pruebas.

--lts-version=versión

```
l versión
```

`versión` es la versión LTS de publicación deseada. Si se omite se usa la última publicación.

--recursive

```
r
```

Recorre el grafo de dependencias del paquete original proporcionado recursivamente y genera expresiones de paquete para todos aquellos paquetes que no estén todavía en Guix.

La siguiente orden importa los metadatos del paquete Haskell HTTP incluido en la versión de publicación LTS de Stackage 7.18:

```
guix import stackage --lts-version=7.18 HTTP
```

**elpa**

Importa metadatos desde el repositorio de archivos de paquetes Emacs Lisp (ELPA) (véase Sección “Packages” en The GNU Emacs Manual).

Las opciones específicas de línea de ordenes son:

--archive=repo

```
a repo
```

`repo` identifica el repositorio de archivos del que obtener la información. Actualmente los repositorios disponibles y sus identificadores son:


Los paquetes de elpa.gnu.org están firmados con una de las claves que contiene el anillo de claves GnuPG en share/emacs/25.1/etc/package-keyring.gpg (o similar) en el paquete emacs (véase Sección “Package Installation” en The GNU Emacs Manual).
- MELPA (https://melpa.org/packages), seleccionado con el identificador melpa.

--recursive

--recursive: Recorre el grafo de dependencias del paquete original proporcionado recursivamente y genera expresiones de paquete para todos aquellos paquetes que no estén todavía en Guix.

crate

Importa metadatos desde el repositorio de paquetes Rust crates.io (https://crates.io), como en este ejemplo:

guix import crate blake2-rfc

El importador de crate también le permite especificar una cadena de versión:

guix import crate constant-time-eq@0.1.0

La opciones adicionales incluyen:

--recursive

--recursive: Recorre el grafo de dependencias del paquete original proporcionado recursivamente y genera expresiones de paquete para todos aquellos paquetes que no estén todavía en Guix.

opam

Importa metadatos desde el repositorio de paquetes OPAM (https://opam.ocaml.org/) usado por la comunidad OCaml.

La estructura del código de guix import es modular. Sería útil tener más importadores para otros formatos de paquetes, y su ayuda es bienvenida aquí (véase Capítulo 14 [Contribuir], página 435).

7.6 Invocación de guix refresh

La principal audiencia de guix refresh son desarrolladoras de la distribución de software GNU. Por defecto, informa de cualquier paquete proporcionado por la distribución que esté anticuado comparado con la última versión oficial, de esta manera:

$ guix refresh

gnu/packages/gettext.scm:29:13: gettext would be upgraded from 0.18.1.1 to 0.18.2.1

gnu/packages/glib.scm:77:12: glib would be upgraded from 2.34.3 to 2.37.0

De manera alternativa, se pueden especificar los paquetes a considerar, en cuyo caso se emite un aviso para paquetes que carezcan de actualizador:

$ guix refresh coreutils guile guile-ssh

gnu/packages/ssh.scm:205:2: warning: no updater for guile-ssh

gnu/packages/guile.scm:136:12: guile would be upgraded from 2.0.12 to 2.0.13

guix refresh navega por los repositorios oficiales de cada paquete y determina el número de versión mayor entre las publicaciones encontradas. La orden sabe cómo actualizar tipos específicos de paquetes: paquetes GNU, paquetes ELPA, etc.—vea la documentación de --type más adelante. Hay muchos paquetes, no obstante, para los que carece de un método
para determinar si está disponible una versión oficial posterior. No obstante, el mecanismo es extensible, ¡no tenga problema en contactar con nosotras para añadir un método nuevo!

```
--recursive

Considera los paquetes especificados, y todos los paquetes de los que dependen.

$ guix refresh --recursive coreutils
gnu/packages/acl.scm:35:2: warning: no updater for acl
gnu/packages/m4.scm:30:12: info: 1.4.18 is already the latest version of m4
gnu/packages/xml.scm:68:2: warning: no updater for expat
gnu/packages/multiprecision.scm:40:12: info: 6.1.2 is already the latest version
```

A veces el nombre oficial es diferente al nombre de paquete usado en Guix, y `guix refresh` necesita un poco de ayuda. La mayor parte de los actualizadores utilizan la propiedad `upstream-name` en las definiciones de paquetes, que puede usarse para obtener dicho efecto:

```
(define-public network-manager
  (package
    (name "network-manager")
    ;; ...
    (properties '((upstream-name . "NetworkManager")))))
```

Cuando se proporciona `--update`, modifica los ficheros de fuentes de la distribución para actualizar los números de versión y hash de los archivadores tar de fuentes en las recetas de los paquetes (véase Sección 6.2 [Definición de paquetes], página 77). Esto se consigue con la descarga del último archivador de fuentes del paquete y su firma OpenPGP asociada, seguida de la verificación del archivador descargado y su firma mediante el uso de `gpg`, y finalmente con el cálculo de su hash—tena en cuenta que GnuPG debe estar instalado y en `$PATH`; ejecute `guix install gnupg` si es necesario.

Cuando la clave pública usada para firmar el archivador no se encuentra en el anillo de claves de la usuaria, se intenta automáticamente su obtención desde un servidor de claves públicas; cuando se encuentra, la clave se añade al anillo de claves de la usuaria; en otro caso, `guix refresh` informa de un error.

Se aceptan las siguientes opciones:

```
--expression=expr -e expr

Considera el paquete al que evalúa expr

Es útil para hacer una referencia precisa de un paquete concreto, como en este ejemplo:

    guix refresh -l -e '(@@ (gnu packages commencement) glibc-final)'

Esta orden enumera los paquetes que dependen de la libc “final” (esencialmente todos los paquetes).
```

```
--update -u

Actualiza los ficheros fuente de la distribución (recetas de paquetes) en su lugar. Esto se ejecuta habitualmente desde una copia de trabajo del árbol de fuentes de Guix (véase Sección 14.2 [Ejecución de Guix antes de estar instalado], página 436):

    $ ./pre-inst-env guix refresh -s non-core -u
```
Véase Sección 6.2 [Definición de paquetes], página 77, para más información sobre la definición de paquetes.

--select=[subconjunto]
-s subconjunto
Selecciona todos los paquetes en subconjunto, o bien core o bien non-core. El subconjunto core hace referencia a todos los paquetes en el núcleo de la distribución—es decir, paquetes que se usan para construir “todo lo demás”. Esto incluye GCC, libc, Binutils, Bash, etc. Habitualmente, cambiar uno de esos paquetes en la distribución conlleva la reconstrucción de todos los demás. Por tanto, esas actualizaciones son una inconveniencia para las usuarias en términos de tiempo de construcción o ancho de banda usado por la actualización.

El subconjunto non-core hace referencia a los paquetes restantes. Es típicamente útil en casos donde una actualización de paquetes básicos no sería conveniente.

--manifest=fichero
-m fichero
Selecciona todos los paquetes del manifiesto en fichero. Es útil para comprobar si algún paquete del manifiesto puede actualizarse.

--type=actualizador
-t actualizador
Selecciona únicamente paquetes manejados por actualizador (puede ser una lista separada por comas de actualizadores). Actualmente, actualizador puede ser:
gnu el actualizador de paquetes GNU;
gnome el actualizador para paquetes GNOME;
kde el actualizador para paquetes KDE;
xorg el actualizador para paquetes X.org;
kernel.org el actualizador para paquetes alojados en kernel.org;
elpa el actualizador para paquetes ELPA (https://elpa.gnu.org/);
cran el actualizador para paquetes CRAN (https://cran.r-project.org/);
bioconductor el actualizador para paquetes R Bioconductor (https://www.bioconductor.org/);
cpan el actualizador para paquetes CPAN (https://www.cpan.org/);
pypi el actualizador para paquetes PyPI (https://pypi.python.org).
gem el actualizador para paquetes RubyGems (https://rubygems.org).
github el actualizador para paquetes GitHub (https://github.com).
Capítulo 7: Utilidades

hackage  el actualizador para paquetes Hackage (https://hackage.haskell.org).
stackage  el actualizador para paquetes Stackage (https://www.stackage.org).
launchpad  el actualizador para paquetes Launchpad (https://launchpad.net).

Por ejemplo, la siguiente orden únicamente comprueba actualizaciones de paquetes Emacs alojados en elpa.gnu.org y actualizaciones de paquetes CRAN:

```bash
$ guix refresh --type=elpa,cran
```

gnu/packages/statistics.scm:819:13: r-testthat would be upgraded from 0.10.0 to 0.11.0

gnu/packages/emacs.scm:856:13: emacs-auctex would be upgraded from 11.88.6 to 11.88.9

Además, guix refresh puede recibir uno o más nombres de paquetes, como en este ejemplo:

```bash
$ ./pre-inst-env guix refresh -u emacs idutils gcc@4.8
```

La orden previa actualiza específicamente los paquetes emacs y idutils. La opción --select no tendría efecto en este caso.

Cuando se considere la actualización de un paquete, a veces es conveniente conocer cuantos paquetes se verían afectados por la actualización y su compatibilidad debería comprobarse. Para ello la siguiente opción puede usarse cuando se proporcionan uno o más nombres de paquete a guix refresh:

```bash
--list-updaters
```

- L Enumera los actualizadores disponibles y finaliza (vea la opción previa --type). Para cada actualizador, muestra la fracción de paquetes que cubre; al final muestra la fracción de paquetes cubiertos por todos estos actualizadores.

```bash
--list-dependent
```

- L Enumera los paquetes de nivel superior dependentes que necesitarían una reconstrucción como resultado de la actualización de uno o más paquetes.

Véase Sección 7.9 [Invocación de guix graph], página 145, para información sobre cómo visualizar la lista de paquetes que dependen de un paquete.

Sea consciente de que la opción --list-dependent únicamente aproxima las reconstrucciones necesarias como resultado de una actualización. Más reconstrucciones pueden ser necesarias bajo algunas circunstancias.

```bash
$ guix refresh --list-dependent flex
```

Building the following 120 packages would ensure 213 dependent packages are rebuilt:

hop@2.4.0  geiser@0.4  notmuch@0.18  mu@0.9.9.5  cflow@1.4  idutils@4.6 ...

La orden previa enumera un conjunto de paquetes que puede ser construido para comprobar la compatibilidad con una versión actualizada del paquete flex.

```bash
--list-transitive
```

Enumera todos los paquetes de los que uno o más paquetes dependen.

```bash
$ guix refresh --list-transitive flex
```
**Capítulo 7: Utilidades**

flex@2.6.4 depends on the following 25 packages: perl@5.28.0 help2man@1.47.6
bison@3.0.5 indent@2.2.10 tar@1.30 gzip@1.9 bzip2@1.0.6 xz@5.2.4 file@5.33

La orden previa enumera un conjunto de paquetes que, en caso de cambiar, causarían la reconstrucción de **flex**.

Las siguientes opciones pueden usarse para personalizar la operación de GnuPG:

**--gpg=orden**
Use *orden* como la orden de GnuPG 2.x. Se busca *orden* en *PATH*.

**--keyring=fichero**
Usa *fichero* como el anillo de claves para claves de proveedoras. *fichero* debe estar en el formato *keybox*. Los ficheros Keybox normalmente tienen un nombre terminado en .kbx y GNU Privacy Guard (GPG) puede manipular estos ficheros (véase Sección “kbxutil” en *Using the GNU Privacy Guard*, para información sobre una herramienta para manipular ficheros keybox).

Cuando se omite esta opción, *guix refresh* usa */.config/guix/upstream/trustedkeys.kbx* como el anillo de claves para las firmas de proveedoras. Las firmas OpenPGP son comprobadas contra claves de este anillo; las claves que falten son descargadas a este anillo de claves también (véase **--key-download** a continuación).

Puede exportar claves de su anillo de claves GPG predeterminado en un fichero keybox usando órdenes como esta:

```
gpg --export rms@gnu.org | kbxutil --import-openpgp >> mianillo.kbx
```

Del mismo modo, puede obtener claves de un archivo keybox específico así:

```
gpg --no-default-keyring --keyring mianillo.kbx \
--recv-keys 3CE464558A84FDC69DB40CFB090B11993D9AEBB5
```

Sección “GPG Configuration Options” en *Using the GNU Privacy Guard*, para más información sobre la opción **--keyring** de GPG.

**--key-download=política**
Maneja las claves no encontradas de acuerdo a la *política*, que puede ser una de:

- **always**: Siempre descarga las claves OpenPGP no encontradas del servidor de claves, y las añade al anillo de claves GnuPG de la usuaria.
- **never**: Nunca intenta descargar claves OpenPGP no encontradas. Simplemente propaga el error.
- **interactive**: Cuando se encuentra un paquete firmado por una clave OpenPGP desconocida, pregunta a la usuaria si descargarla o no. Este es el comportamiento predeterminado.

**--key-server=dirección**
Use *dirección* como el servidor de claves OpenPGP cuando se importa una clave pública.
--load-path=directorio

Añade directorio al frente de la ruta de búsqueda de módulos de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76).

Esto permite a las usuarias definir sus propios paquetes y hacerlos visibles a las herramientas de línea de órdenes.

El actualizador github usa el API GitHub (https://developer.github.com/v3/) para consultar nuevas publicaciones. Cuando se usa repetidamente, por ejemplo al comprobar todos los paquetes, GitHub terminará rechazando las peticiones siguientes a través de su API. Por defecto se permiten 60 peticiones por hora a través de su API, y una actualización completa de todos los paquetes de GitHub en Guix necesita más que eso. La identificación con GitHub a través del uso de un identificador de su API (token) amplía esos límites. Para usar un identificador API, establezca la variable de entorno GUIX_GITHUB_TOKEN al identificador proporcionado por https://github.com/settings/tokens u otra fuente.

7.7 Invocación de guix lint

La orden guix lint sirve para ayudar a las desarrolladoras de paquetes a evitar errores comunes y usar un estilo consistente. Ejecuta un número de comprobaciones en un conjunto de paquetes proporcionado para encontrar errores comunes en sus definiciones. Las comprobaciones disponibles incluyen (véase --list-checkers para una lista completa):

- synopsis
  Valida ciertas reglas tipográficas y de estilo en la descripción y sinopsis de cada paquete.

- description
  Identifica entradas que probablemente deberían ser entradas nativas.

- inputs-should-be-native
  Identifica entradas que probablemente deberían ser entradas nativas.

- source
  home-page
  mirror-url
  github-url
  source-file-name
  Comprueba las URL home-page y source e informa aquellas que no sean válidas. Sugiere una URL mirror:// cuando sea aplicable. Si la URL source redirecciona a una URL GitHub, recomienda el uso de la URL GitHub. Comprueba que el nombre de fichero de las fuentes es significativo, por ejemplo que no es simplemente un número de versión o revisión git, sin un nombre file-name declarado (véase sección 6.2.2 [Referencia de origin], página 83).

- source-unstable-tarball
  Analiza la URL source para determinar si un archivador tar de GitHub se genera de forma automática o es una publicación oficial. Desafortunadamente los archivadores tar de GitHub a veces se regeneran.

- archival
  Comprueba si el código fuente del paquete se encuentra archivado en Software Heritage (https://www.softwareheritage.org).
Cuando el código fuente que no se encuentra archivado proviene de un sistema de control de versiones\(^2\)—por ejemplo, se ha obtenido con \texttt{git-fetch}—, envía a Software Heritage una petición de almacenamiento de manera que se archive cuando sea posible. Esto asegura que las fuentes permanecen disponibles a largo plazo, y que Guix puede usar Software Heritage como respaldo en caso de que el código fuente desapareciera de la máquina que lo almacenaba originalmente. El estado de las peticiones de almacenamiento recientes puede verse en su página web (https://archive.softwareheritage.org/save/#requests).

Cuando el código fuente es un fichero comprimido que se obtiene con \texttt{url-fetch}, simplemente imprime un mensaje cuando no se encuentra archivado. En el momento de la escritura de este documento, Software Heritage no permite el almacenamiento de ficheros comprimidos arbitrarios; estamos trabajando en formas de asegurar que también se archive el código que no se encuentra bajo control de versiones.

Software Heritage limita la tasa de peticiones por dirección IP (https://archive.softwareheritage.org/api/#rate-limiting). Cuando se alcanza dicho límite, \texttt{guix lint} imprime un mensaje y la comprobación \texttt{archival} no hace nada hasta que dicho límite se reinicie.

cve

Informa de vulnerabilidades encontradas en las bases de datos de vulnerabilidades y exposiciones comunes (CVE) del año actual y el pasado publicadas por el NIST de EEUU (https://nvd.nist.gov/vuln/data-feeds).

Para ver información acerca de una vulnerabilidad particular, visite páginas como:

- ‘https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-YYYY-ABCD’

donde CVE-YYYY-ABCD es el identificador CVE—por ejemplo, CVE-2015-7554. Las desarrolladoras de paquetes pueden especificar en las recetas del paquete el nombre y versión en la plataforma común de enumeración (CPE) (https://nvd.nist.gov/cpe.cfm) del paquete cuando el nombre o versión que usa Guix son diferentes, como en este ejemplo:

```guile
(package
 (name "grub")
 ;; ...
 ;; CPE llama a este paquete "grub2".
 (properties '((cpe-name . "grub2")
               (cpe-version . "2.3")))
```

Algunas entradas en la base de datos CVE no especifican a qué versión del paquete hacen referencia, y por lo tanto “permanecen visibles” para siempre. Las desarrolladoras de paquetes que encuentren alertas CVE y verifiquen que pueden ignorarse, pueden declararlas como en este ejemplo:

```guile
(package
 (name "t1lib")
```

\(^2\) VCS en inglés
Estas alertas de CVE no aplican y pueden ignorarse con seguridad.

(properties '(lint-hidden-cve . ("CVE-2011-0433" "CVE-2011-1553" "CVE-2011-1554" "CVE-2011-5244")))

formatting

Avisa de problemas de formato obvios en el código fuente: espacios en blanco al final de las líneas, uso de tabuladores, etc.

La sintaxis general es:

```
guix lint opciones paquete...
```

Si no se proporciona ningún paquete en la línea de órdenes, todos los paquetes se com-probaran. Las opciones pueden ser cero o más de las siguientes:

--list-checkers
    Enumera y describe todas las comprobaciones disponibles que se ejecutarán sobre los paquetes y finaliza.

--checkers
    Únicamente activa las comprobaciones especificadas en una lista separada por comas que use los nombres devueltos por --list-checkers.

--load-path=directorio
    Añade directorio al frente de la ruta de búsqueda de módulos de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76).
    Esto permite a las usuarias definir sus propios paquetes y hacerlos visibles a las herramientas de línea de órdenes.

7.8 Invocación de guix size

La orden `guix size` ayuda a las desarrolladoras de paquetes a perfilar el uso de disco de los paquetes. Es fácil pasar por encima el impacto que produce añadir una dependencia adicional a un paquete, o el impacto del uso de una salida única para un paquete que puede ser dividido fácilmente (véase Sección 4.4 [Paquetes con múltiples salidas], página 47). Estos son los problemas típicos que `guix size` puede resaltar.

Se le pueden proporcionar una o más especificaciones de paquete como `gcc@4.8` o `guile:debug`, o un nombre de fichero en el almacén. Considere este ejemplo:

```
$ guix size coreutils
store item total  self
/gnu/store/...-gcc-5.5.0-lib 60.4  30.1 38.1%
/gnu/store/...-glibc-2.27 30.3  28.8 36.6%
/gnu/store/...-coreutils-8.28 78.9  15.0 19.0%
/gnu/store/...-gmp-6.1.2 63.1  2.7  3.4%
/gnu/store/...-bash-static-4.4.12 1.5  1.5  1.9%
/gnu/store/...-acl-2.2.52 61.1  0.4  0.5%
```
Capítulo 7: Utilidades

Los elementos del almacén enumerados aquí constituyen la clausura transitiva de Coreutils—es decir, Coreutils y todas sus dependencias, recursivamente—como sería devuelto por:

$ guix gc -R /gnu/store/...-coreutils-8.23

Aquí la salida muestra tres columnas junto a los elementos del almacén. La primera columna, etiquetada “total”, muestra el tamaño en mebibytes (MiB) de la clausura del elemento del almacén—es decir, su propio tamaño sumado al tamaño de todas sus dependencias. La siguiente columna, etiquetada “self”, muestra el tamaño del elemento en sí. La última columna muestra la relación entre el tamaño del elemento en sí frente al espacio ocupado por todos los elementos enumerados.

En este ejemplo, vemos que la clausura de Coreutils ocupa 79 MiB, cuya mayor parte son libc y las bibliotecas auxiliares de GCC para tiempo de ejecución. (Que libc y las bibliotecas de GCC representen una fracción grande de la clausura no es un problema en sí, puesto que siempre están disponibles en el sistema de todas maneras).

Cuando los paquetes pasados a guix size están disponibles en el almacén3 consultando al daemon para determinar sus dependencias, y mide su tamaño en el almacén, de forma similar a du -ms --apparent-size (véase Sección “du invocation” en GNU Coreutils).

Cuando los paquetes proporcionados no están en el almacén, guix size informa en base de las sustituciones disponibles (véase Sección 4.3 [Sustituciones], página 44). Esto hace posible perfilar el espacio en disco incluso de elementos del almacén que no están en el disco, únicamente disponibles de forma remota.

Puede especificar también varios nombres de paquetes:

$ guix size coreutils grep sed bash

En este ejemplo vemos que la combinación de los cuatro paquetes toma 102.3 MiB en total, lo cual es mucho menos que la suma de cada clausura, ya que tienen muchas dependencias en común.

Las opciones disponibles son:

--substitute-urls=urls

Usa la información de sustituciones de urls. Véase [client-substitute-urls], página 120.

3 Más precisamente, guix size busca la variante sin injertos de los paquetes, como el devuelto por guix build paquete --no-grafts. Véase Capítulo 11 [Actualizaciones de seguridad], página 425, para información sobre injertos.
Capítulo 7: Utilidades

--sort=clave

Ordea las líneas de acuerdo a clave, una de las siguientes opciones:

self el tamaño de cada elemento (predeterminada);
clausura el tamaño total de la clausura del elemento.

--map-file=fichero

Escribe un mapa gráfico del uso del disco en formato PNG en el fichero.
Para el ejemplo previo, el mapa tiene esta pinta:

Esta opción necesita que la biblioteca Guile-Charting (https://wingolog.org/software/guile-charting/) esté instalada y visible en la ruta de búsqueda de módulos Guile. Cuando no es el caso, guix size produce un error al intentar cargarla.

--system=sistema
-s sistema

Considera paquetes para sistema—por ejemplo, x86_64-linux.

--load-path=directório
-L directório

Añade directório al frente de la ruta de búsqueda de módulos de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76).

Esto permite a las usuarias definir sus propios paquetes y hacerlos visibles a las herramientas de línea de órdenes.

7.9 Invocación de guix graph

Los paquetes y sus dependencias forman un grafo, específicamente un grafo acíclico dirigido (GAD, DAG en Inglés). Puede hacerse difícil rápidamente tener un modelo mental del GAD del paquete, por lo que la orden guix graph proporciona una representación virtual del GAD. Por defecto, guix graph emite una representación en GAD en el formato de entrada de Graphviz (https://graphviz.org/), por lo que su salida puede ser pasada directamente a la herramienta dot de Graphviz. También puede emitir una página HTMP con código JavaScript embebido para mostrar un diagrama de cuerdas en un navegador Web, usando la biblioteca d3.js (https://d3js.org/), o emitir consultas Cypher para construir un grafo en una base de datos de grafos que acepte el lenguaje de consultas openCypher (https://www.opencypher.org/). La sintaxis general es:

guix graph opciones paquete...
Por ejemplo, la siguiente orden genera un fichero PDF que representa el GAD para GNU Core Utilities, mostrando sus dependencias en tiempo de construcción:

```
guix graph coreutils | dot -Tpdf > gad.pdf
```

La salida es algo así:

```
coreutils-8.25
perl-5.24.0
acl-2.2.52 gmp-6.1.1 libcap-2.24
gettext-minimal-0.19.8.1
attr-2.4.47 m4-1.4.17
expat-2.2.0
```

Bonito y pequeño grafo, ¿no?

¡Pero hay más de un grafo! El grafo previo es conciso: es el grafo de los objetos package, omitiendo las entradas implícitas como GCC, libc, grep, etc. Es habitualmente útil tener un grafo conciso así, pero a veces una puede querer ver más detalles. `guix graph` implementa varios tipos de grafos, lo que le permite seleccionar el nivel de detalle:

- **package**: Este es el tipo por defecto usado en el ejemplo previo. Muestra el GAD de objetos package, excluyendo dependencias implícitas. Es conciso, pero deja fuera muchos detalles.
- **reverse-package**: Esto muestra el GAD inverso de paquetes. Por ejemplo:

```
guix graph --type=reverse-package ocaml
```

... emite el grafo de paquetes que dependen _explicitamente_ de OCaml (si también tiene interés en casos donde OCaml es una dependencia implícita, véase `reverse-bag` a continuación).

Fíjese que esto puede producir grafos inmensos para los paquetes básicos. Si todo lo que quiere saber es el número de paquetes que dependen de uno determinado, use `guix refresh --list-dependent` (véase Sección 7.6 [Invocación de guix refresh], página 136).

- **bag-emerged**: Este es el GAD del paquete, _incluyendo_ entradas implícitas.

Por ejemplo, la siguiente orden:

```
guix graph --type=bag-emerged coreutils | dot -Tpdf > gad.pdf
```
... emite este grafo más grande:

En la parte inferior del grafo, vemos todas las entradas implícitas de *gnu-build-system* (véase Sección 6.3 [Sistemas de construcción], página 85).

Ahora bien, fíjese que las dependencias de estas entradas implícitas—es decir, las *dependencias del lanzamiento inicial* (véase Capítulo 12 [Lanzamiento inicial], página 427)—no se muestran aquí para mantener una salida concisa.

**bag**

Similar a *bag-emerged*, pero esta vez incluye todas las dependencias del lanzamiento inicial.

**bag-with-origins**

Similar a *bag*, pero también muestra los orígenes y sus dependencias.

**reverse-bag**

Muestra el GAD *inverso* de paquetes. Al contrario que *reverse-package*, también tiene en cuenta las dependencias implícitas. Por ejemplo:

```
guix graph -t reverse-bag dune
```

... emite el grafo de todos los paquetes que dependen de Dune, directa o indirectamente. Ya que Dune es una dependencia *implícita* de muchos paquetes *vía*
dune-build-system, esto mostrará un gran número de paquetes, mientras que reverse-package mostraría muy pocos si muestra alguno.

derivación
Esta es la representación más detallada: muestra el GAD de derivaciones (véase Sección 6.5 [Derivaciones], página 101) y elementos simples del almacén. Comparada con las representaciones previas, muchos nodos adicionales son visibles, incluyendo los guiones de construcción, parches, módulos Guile, etc.
Para este tipo de grafo, también es posible pasar un nombre de fichero .drv en vez del nombre del paquete, como en:

```bash
guix graph -t derivation 'guix system build -d mi-configuración.scm'
```

module
Este es el grafo de los módulos de paquete (véase Sección 6.1 [Módulos de paquetes], página 76). Por ejemplo, la siguiente orden muestra el grafo para el módulo de paquetes que define el paquete guile:

```bash
guix graph -t module guile | dot -Tpdf > grafo-del-modulo.pdf
```

Todos los tipos previos corresponden a las dependencias durante la construcción. El grafo siguiente representa las dependencias en tiempo de ejecución:

references
Este es el grafo de referencias de la salida de un paquete, como lo devuelve guix gc --references (véase Sección 4.5 [Invocación de guix gc], página 48).
Si la salida del paquete proporcionado no está disponible en el almacén, guix graph intenta obtener la información de dependencias desde las sustituciones. Aquí también puede proporcionar un nombre de fichero del almacén en vez de un nombre de paquete. Por ejemplo, la siguiente orden produce el grafo de referencias de su perfil (¡el cuál puede ser grande!):

```bash
guix graph -t references 'readlink -f ~/.guix-profile'
```

referrers
Este es el grafo de referentes de la salida de un paquete, como lo devuelve guix gc --referrers (véase Sección 4.5 [Invocación de guix gc], página 48).
Depende exclusivamente de información en su almacén. Por ejemplo, supongamos que la versión actual de Inkscape está disponible en 10 perfiles en su máquina; guix graph -t referrers inkscape mostrará un grafo cuya raíz es Inkscape y con esos 10 perfiles enlazados a ella.
Puede ayudar a determinar qué impide que un elemento del almacén sea recolectado.

Las opciones disponibles son las siguientes:

--type=tipo
-t tipo Produce un grafo de salida de tipo, donde tipo debe ser uno de los valores enumerados previamente.

--list-types
Enumera los tipos de grafos implementados.

--backend=motor
-b motor Produce un grafo usando el motor seleccionado.
Capítulo 7: Utilidades

--list-backends
Enumera los motores de grafos implementados.
Actualmente, los motores disponibles son Graphviz y d3.js.

--expression=expr
-e expr
Considera el paquete al que evalúa expr
Es útil para hacer una referencia precisa de un paquete concreto, como en este ejemplo:

```
guix graph -e '@@ (gnu packages commencement) gnu-make-final')'
```

--system=sistema
-s sistema
Muestra el grafo para sistema—por ejemplo, i686-linux.
El grafo de dependencias del paquete es altamente independiente de la arquitectura, pero existen algunas partes dependientes de la arquitectura que esta opción le permite visualizar.

--load-path=directorio
-L directorio
Añade directorio al frente de la ruta de búsqueda de módulos de paquetes (véase Sección 6.1 [Módulos de paquetes], página 76).
Esto permite a las usuarias definir sus propios paquetes y hacerlos visibles a las herramientas de línea de órdenes.

Además de esto, guix graph permite todas las opciones habituales de transformación de paquetes (véase Sección 7.1.2 [Opciones de transformación de paquetes], página 122). Esto facilita la visualización del efecto de una transformación de reescritura de grafo como --with-input. Por ejemplo, la siguiente orden muestra el grafo de git una vez que openssl ha sido reemplazado por libressl en todos los nodos del grafo:

```
guix graph git --with-input=openssl=libressl
```

¡Tantas posibilidades, tanta diversión!

7.10 Invocación de guix publish
El propósito de guix publish es permitir a las usuarias compartir fácilmente su almacén con otras, quienes pueden usarlo como servidor de sustituciones (véase Sección 4.3 [Sustituciones], página 44).

Cuando guix publish se ejecuta, lanza un servidor HTTP que permite a cualquiera que tenga acceso a través de la red obtener sustituciones de él. Esto significa que cualquier máquina que ejecute Guix puede actuar como si fuese una granja de construcción, ya que la interfaz HTTP es compatible con Cuirass, el software detrás de la granja de construcción ci.guix.gnu.org.

Por seguridad, cada sustitución se firma, permitiendo a las receptoras comprobar su autenticidad e integridad (véase Sección 4.3 [Sustituciones], página 44). Debido a que guix publish usa la clave de firma del sistema, que es únicamente legible por la administradora del sistema, debe iniciarse como root; la opción --user hace que renuncie a sus privilegios tan pronto como sea posible.
El par claves de firma debe generarse antes de ejecutar `guix publish`, usando `guix archive --generate-key` (véase Sección 4.11 [Invocación de guix archive], página 63).

La sintaxis general es:

```
guix publish opciones...
```

La ejecución de `guix publish` sin ningún parámetro adicional lanzará un servidor HTTP en el puerto 8080:

```
guix publish
```

Una vez el servidor de publicación ha sido autorizado (véase Sección 4.11 [Invocación de guix archive], página 63), el daemon puede descargar sustituciones de él:

```
guix-daemon --substitute-urls=http://example.org:8080
```

Por defecto, `guix publish` comprime los archivos al vuelo cuando es necesario. Este modo “al vuelo” es conveniente ya que no necesita configuración y está disponible inmediatamente. No obstante, cuando se proporciona servicio a muchos clientes, se recomienda usar la opción `--cache`, que activa el almacenamiento en caché de los archivos antes de enviarlos a los clientes—véase a continuación para más detalles. La orden `guix weather` proporciona una forma fácil de comprobar lo que proporciona un servidor (véase Sección 7.14 [Invocación de guix weather], página 157).

Además `guix publish` también sirve como un espejo de acceso por contenido a ficheros de fuentes a los que los registros `origin` hacen referencia (véase Sección 6.2.2 [Referencia de origen], página 83). Por ejemplo, si asumimos que `guix publish` se ejecuta en `example.org`, la siguiente URL devuelve directamente el fichero `hello-2.10.tar.gz` con el hash SHA256 proporcionado (representado en formato `nix-base32`, véase Sección 7.4 [Invocación de guix hash], página 130).

```
http://example.org/file/hello-2.10.tar.gz/sha256/0ssi1...ndq1i
```

Obviamente estas URL funcionan solamente para ficheros que se encuentran en el almacén; en otros casos devuelven un 404 (“No encontrado”).

Los log de construcción están disponibles desde URL `/log` como:

```
http://example.org/log/gwspk...-guile-2.2.3
```

Cuando `guix-daemon` está configurado para almacenar comprimidos los log de construcción, como sucede de forma predeterminada (véase Sección 2.5 [Invocación de guix-daemon], página 15), las URL `/log` devuelven los log igualmente comprimidos, con un `Content-Type` adecuado y/o una cabecera `Content-Encoding`. Recomendamos ejecutar `guix-daemon` con `--log-compression=gzip` ya que los navegadores Web pueden extraer el contenido automáticamente, lo cual no es el caso con la compresión bzip2.

Las siguientes opciones están disponibles:

```
--port=puerto  
-p puerto     Escucha peticiones HTTP en puerto.

--listen=dirección  
    Escucha en la interfaz de red de la dirección. El comportamiento predeterminado es aceptar conexiones de cualquier interfaz.
```
--user=usuaria
-u usuaria
Cambia los privilegios a los de usuaria tan pronto como sea posible—es decir, una vez el socket del servidor esté abierto y la clave de firma haya sido leída.

--compression[=método[:nivel]]
-C [método[:nivel]]
Comprime los datos usando el método y nivel proporcionados. método es o bien lzip o bien gzip; cuando método se omite, se usa gzip.
Cuando el nivel es cero, desactiva la compresión. El rango 1 a 9 corresponde a distintos niveles de compresión gzip: 1 es el más rápido, y 9 es el mejor (intensivo a nivel de CPU). El valor predeterminado es 3.
Habitualmente lzip comprime notablemente mejor que gzip a cambio de un pequeño incremento en el uso del procesador; véase las pruebas en la página web de lzip (https://nongnu.org/lzip/lzip_benchmark.html).
A menos que se use --cache, la compresión ocurre al vuelo y los flujos comprimidos no se almacenan en caché. Por tanto, para reducir la carga en la máquina que ejecuta guix publish, puede ser una buena idea elegir un nivel de compresión bajo, ejecutar guix publish detrás de un proxy con caché o usar --cache. El uso de --cache tiene la ventaja de que permite a guix publish añadir la cabecera HTTP Content-Length a sus respuestas.
Se puede repetir esta opción, en cuyo caso cada sustitución se comprime usando todos los métodos seleccionados, y todos son anunciados. Esto es útil cuando las usuarias pueden no implementar todos los métodos de compresión: pueden seleccionar el que implementan.

--cache=directorio
-c directorio
Almacena en caché los archivos y metadatos (URL .narinfo) en directorio y únicamente proporciona archivos que están en la caché.
Cuando se omite esta opción, los archivos y metadatos se crean al vuelo. Esto puede reducir el ancho de banda disponible, especialmente cuando la compresión esté activa, ya que se puede llegar al límite de la CPU. Otra desventaja del modo predeterminado es que la longitud de los archivos no se conoce con anterioridad, por lo que guix publish no puede añadir la cabecera HTTP Content-Length a sus respuestas, lo que a su vez previene que los clientes conozcan la cantidad de datos a descargar.
De manera contraria, cuando se usa --cache, la primera petición de un elemento del almacén (a través de una URL .narinfo) devuelve 404 e inicia un proceso en segundo plano para cocinar el archivo—calcular su .narinfo y comprimirla, en caso necesario. Una vez el archivo está alojado en la caché de directorio, las siguientes peticiones obtendrán un resultado satisfactorio y se ofrecerá el contenido directamente desde la caché, lo que garantiza que los clientes obtienen el mejor ancho de banda posible.
El proceso de “cocinado” se realiza por hilos de trabajo. Por defecto, se crea un hilo por núcleo de la CPU, pero puede ser personalizado. Véase --workers a continuación.
Cuando se usa --ttl, las entradas en caché se borran automáticamente cuando hayan expirado.

--workers=N
Cuando se usa --cache, solicita la creación de N hilos de trabajo para “cocinar” archivos.

--ttl=ttl
Produce cabeceras HTTP Cache-Control que anuncian un tiempo-de-vida (TTL) de ttl. ttl debe indicar una duración: 5d significa 5 días, 1m significa un mes, etc.

Esto permite a la usuaria de Guix mantener información de sustituciones en la caché durante ttl. No obstante, fíjese que guix publish no garantiza en sí que los elementos del almacén que proporciona de hecho permanezcan disponibles hasta que ttl expire.

Adicionalmente, cuando se usa --cache, las entradas en caché que no hayan sido accedidas en ttl y no tengan un elemento correspondiente en el almacén pueden ser borradas.

--nar-path=ruta
Usa ruta como el prefijo para las URL de los archivos “nar” (véase Sección 4.11 [Invocación de guix archive], página 63).

Por defecto, los archivos nar se proporcionan en una URL como /nar/gzip/...-coreutils-8.25. Esta opción le permite cambiar la parte /nar por ruta.

--public-key=fichero
--private-key=fichero
Usa los ficheros específicos como el par de claves pública y privada usados para firmar los elementos del almacén publicados.

Los ficheros deben corresponder al mismo par de claves (la clave privada se usa para la firma y la clave pública simplemente se anuncia en los metadatos de la firma). Deben contener claves en el formato canónico de expresiones-S como el producido por guix archive --generate-key (véase Sección 4.11 [Invocación de guix archive], página 63). Por defecto, se usan /etc/guix/signing-key.pub y /etc/guix/signing-key.sec.

--repl[=puerto]
-r [puerto]

Activar guix publish en el sistema Guix consiste en solo una línea: simplemente instancie un servicio guix-publish-service-type en el campo services del sistema operativo operating-system (véase guix-publish-service-type, página 195)

Si en vez de eso ejecuta Guix en una distribución distinta, siga estas instrucciones:

- Si su distribución anfitriona usa el sistema de inicio systemd:
  
  ```
  # ln -s ~root/.guix-profile/lib/systemd/system/guix-publish.service \
  ```

  ```


```
/etc/systemd/system/
  # systemctl start guix-publish && systemctl enable guix-publish
```

- Si su distribución anfitriona usa el sistema de inicio Upstart:
  ```
  # ln -s ~/root/.guix-profile/lib/upstart/system/guix-publish.conf /etc/init/
  # start guix-publish
  ```
- En otro caso, proceda de forma similar con el sistema de inicio de su distribución.

### 7.11 Invocación de guix challenge

¿Los binarios que proporciona este servidor realmente corresponden al código fuente que dice construir? ¿Es determinista el proceso de construcción de un paquete? Éstas son las preguntas que la orden `guix challenge` intenta responder.

La primera es obviamente una cuestión importante: antes de usar un servidor de sustituciones (véase Sección 4.3 [Sustituciones], página 44), es importante haber `verificado` que proporciona los binarios correctos, y por tanto `ponerlo a prueba`

La segunda es lo que permite la primera: si las construcciones de los paquetes son deterministas, construcciones independientes deberían emitir el mismo resultado, bit a bit; si el servidor proporciona un binario diferente al obtenido localmente, o bien está corrompido o bien tiene intenciones perniciosas.

Sabemos que el hash que se muestra en los nombres de fichero en `/gnu/store` es el hash de todas las entradas del proceso que construyó el fichero o directorio—compiladores, bibliotecas, guiones de construcción, etc. (véase Capítulo 1 [Introducción], página 2). Asumiendo procesos de construcción deterministas, un nombre de fichero del almacén debe corresponder exactamente a una salida de construcción. `guix challenge` comprueba si existe, realmente, una asociación unívoca comparando la salida de la construcción de varias construcciones independientes de cualquier elemento del almacén proporcionado.

La salida de la orden muestra algo así:

```
actualizando sustituciones desde 'https://ci.guix.gnu.org'... 100.0%
actualizando sustituciones desde 'https://guix.example.org'... 100.0%
el contenido de /gnu/store/...-openssl-1.0.2d es diferente:
  hash local: 0725l22r5jnzazaacncwsvp9kgf42266ayyp814v7djsx7nk963q
  https://ci.guix.gnu.org/nar/...-openssl-1.0.2d: 0725l22r5jnzazaacncwsvp9kgf42266ayyp814v7djsx7nk963q
  https://guix.example.org/nar/...-openssl-1.0.2d: 1zy4fmaaqcnjrzrazjkdnsf5gmjxk754b43qkq47llbyak9z0qqyin
ficheros diferentes:
  /lib/libcrypto.so.1.1
  /lib/libssl.so.1.1

e10 el contenido de /gnu/store/...-git-2.5.0 es diferente:
  hash local: 00p3bmryhjxrhpn2gxs2f7h0a151nin05197205pgbk5ra395hayha
  https://ci.guix.gnu.org/nar/...-git-2.5.0: 069nb85bv4d4a6s1rwjdy8v1cn4cwapm3kdbmy81d6zckj3nq9f
  https://guix.example.org/nar/...-git-2.5.0: 0mdqa9w1p6cml6976v4w0sw9r4p5prkj71zfd1877wkl1cq73
fichero diferente:
  /libexec/git-core/git-fsck

e10 el contenido de /gnu/store/...-pius-2.1.1 es diferente:
  hash local: 0k4v3m9z1zp8zzip8zzzizb7d8skj72f9172xv078aq4w173vnq9ig3ax
  https://ci.guix.gnu.org/nar/...-pius-2.1.1: 0k4v3m9z1zp8zzzizb7d8skj72f9172xv078aq4w173vnq9ig3ax
```

4 NdT: challenge en inglés.
En este ejemplo, guix challenge primero recorre el almacen para determinar el conjunto de derivaciones construidas localmente—en oposición a elementos del almacen que fueron descargados de un servidor de sustituciones—y consulta a todos los servidores de sustituciones. Una vez hecho informa de los elementos del almacen para los cuales los servidores obtuvieron un resultado diferente de el obtenido en la construcción local.

Como un ejemplo, guix.example.org siempre obtiene una respuesta diferente. Por otro modo, ci.guix.gnu.org coincide con las construcciones locales, excepto en el caso de Git. Esto puede indicar que el proceso de construcción de Git no es determinista, lo que significa que su salida varía en función de varias cosas que Guix no controla completamente, aunque la construcción de paquetes se realice en entornos aislados (véase Sección 4.1 [Características], página 35). Las fuentes más comunes de indeterminismo incluyen la adición de marcas de tiempo en los resultados de la construcción, la inclusión de números aleatorios y las enumeraciones de directorios ordenadas por número de nodos-i. Véase https://reproducible-builds.org/docs/ para más información.

Para encontrar cuál es el problema con este binario Git, la aproximación más fácil es ejecutar:

```bash
guix challenge git \
  --diff=diffoscope \
  --substitute-urls="https://ci.guix.gnu.org https://guix.example.org"
```

Esto invoca automáticamente diffoscope, que muestra información detallada sobre los ficheros que son diferentes.

De manera alternativa, se puede hacer algo parecido a esto (véase Sección 4.11 [Invocación de guix archive], página 63):

```bash
$ wget -q -O - https://ci.guix.gnu.org/nar/...-git-2.5.0 \
  | guix archive -x /tmp/git
$ diff -ur --no-dereference /gnu/store/...-git.2.5.0 /tmp/git
```

Esta orden muestra la diferencia entre los ficheros resultantes de la construcción local y los ficheros resultantes de la construcción en ci.guix.gnu.org (véase Sección “Overview” en Comparing and Merging Files). La orden diff funciona muy bien en ficheros de texto. Cuando son binarios los ficheros diferentes, una opción mejor es Diffoscope (https://diffoscope.org/), una herramienta que ayuda en la visualización de diferencias en todo tipo de ficheros.

Una vez haya realizado este trabajo, puede determinar si las diferencias son debidas a un procedimiento de construcción no-determinista o a un servidor con intenciones ocultas. Intentamos duramente eliminar las fuentes de indeterminismo en los paquetes para facilitar la verificación de sustituciones, pero por supuesto es un proceso que implica no solo a Guix,
sino a una gran parte de la comunidad del software libre. Entre tanto, **guix challenge** es una herramienta para ayudar a afrontar el problema.

Si está escribiendo paquetes para Guix, le recomendamos que compruebe si ci.guix.gnu.org y otros servidores de sustituciones obtienen el mismo resultado de construcción que el obtenido por usted:

\[
$ \text{guix challenge paquete}
\]
donde *paquete* es una especificación de paquete como `guile@2.0` o `glibc:debug`.

La sintaxis general es:

\[
\text{guix challenge opciones [paquetes... ]}
\]

Cuando se encuentra una diferencia entre el hasheo de un elemento construido localmente y el proporcionado por un servidor de sustituciones; o entre las sustituciones proporcionadas por distintos servidores, esto es mostrado como en el ejemplo previo y el valor de salida es 2 (otros valores no-cero de la salida denotan otros tipos de error).

La única opción de importancia es:

**--substitute-urls=** *urls*

Considera *urls* la lista separada por espacios de URL de fuentes de sustituciones con las que realizar la comparación.

**--diff=** *modo*

Muestra las diferencias encontradas de acuerdo con *modo*, uno de los siguientes:

- **simple** (el predeterminado)
  Muestra la lista de ficheros que son diferentes.

- **diffoscope**
  Cuando *orden* es una ruta absoluta, ejecuta *orden* en vez de Diffoscope.

- **none**
  No muestra más detalles sobre las diferencias.

Por tanto, a menos que se proporcione **--diff=none**, **guix challenge** descarga los contenidos de los elementos del almacén de los servidores de sustituciones proporcionados para poder compararlos.

**--verbose**

**-v**

Muestra detalles sobre coincidencias (contenidos idénticos) además de información sobre las discrepancias.

### 7.12 Invocación de guix copy

La orden **guix copy** copia elementos del almacén de una máquina a la otra a través de una conexión de shell seguro (SSH)\(^5\). Por ejemplo, la siguiente orden copia el paquete **coreutils**, \(^5\) Esta orden únicamente está disponible cuando ha encontrado Guile-SSH. Véase Sección 2.2 [Requisitos], página 7, para detalles.
el perfil de la usuaria y todas sus dependencias a dirección, ingresando en el sistema como usuaria:

```bash
guix copy --to=usuaria@dirección \ coreutils ‘readlink -f ~/.guix-profile’
```

Si alguno de los elementos del almacén a copiar ya están presentes en dirección, no se envían realmente.

La siguiente orden obtiene libreoffice y gimp de dirección, asumiendo que estén disponibles allí:

```bash
guix copy --from=dirección libreoffice gimp
```

La conexión SSH se establece usando el cliente Guile-SSH, que es compatible con OpenSSH: tiene en cuenta `~/.ssh/known_hosts` y `~/.ssh/config`, y usa el agente SSH para la identificación.

La clave usada para firmar los elementos enviados debe estar aceptada por la máquina remota. Del mismo modo, la clave usada por la máquina remota para firmar los elementos recibidos debe estar en `/etc/guix/acl` de modo que sea aceptada por su propio daemon. Véase Sección 4.11 [Invocación de guix archive], página 63, para más información sobre la verificación de elementos del almacén.

La sintaxis general es:

```bash
guix copy [--to=spec|--from=spec] elementos...
```

Siempre debe especificar una de las siguientes opciones:

```bash
--to=spec
--from=spec
```

Especifica la máquina a la que mandar o desde la que recibir. `spec` debe ser una especificación SSH como `example.org`, `carlos@example.org`, o `carlos@example.org:2222`.

Los `elementos` pueden ser tanto nombres de paquetes, como `gimp`, como elementos del almacén, como `/gnu/store/...-idutils-4.6`.

Cuando se especifica el nombre del paquete a enviar, primero se construye si es necesario, a menos que se use `--dry-run`. Se aceptan las opciones comunes de construcción (véase Sección 7.1.1 [Opciones comunes de construcción], página 119).

### 7.13 Invocación de guix container

**Nota:** En la versión 1.0.1, esta herramienta es experimental. La interfaz está sujeta a cambios radicales en el futuro.

El propósito de `guix container` es la manipulación de procesos en ejecución dentro de entornos aislados, normalmente conocido como un “contenedor”, típicamente creado por las órdenes `guix environment` (véase Sección 5.1 [Invocación de guix environment], página 66) y `guix system container` (véase Sección 8.14 [Invocación de guix system], página 399).

La sintaxis general es:

```bash
guix container acción opciones...
```

`acción` especifica la operación a realizar con el contenedor, y `opciones` especifica los parámetros específicos del contexto para la acción.
Las siguientes acciones están disponibles:

**exec** Ejecute una orden en el contexto de un contenedor en ejecución.

La sintaxis es:

```
  guix container exec pid programa parámetros...
```

*pid* especifica el ID del proceso del contenedor en ejecución. *programa* especifica el nombre del fichero ejecutable dentro del sistema de ficheros raíz del contenedor. *parámetros* son opciones adicionales que se pasarán a *programa*.

La siguiente orden lanza un shell interactivo de ingreso al sistema dentro de un contenedor del sistema, iniciado por *guix system container*, y cuyo ID de proceso es 9001:

```
  guix container exec 9001 /run/current-system/profile/bin/bash --login
```

Fíjese que el *pid* no puede ser el proceso creador del contenedor. Debe ser el PID 1 del contenedor o uno de sus procesos hijos.

### 7.14 Invocación de *guix weather*

De manera ocasional tendrá un mal día al no estar las sustituciones disponibles y le toque construir los paquetes a usted misma (véase Sección 4.3 [Sustituciones], página 44). La orden *guix weather* informa de la disponibilidad de sustituciones en los servidores especificados de modo que pueda tener una idea sobre cómo será su día hoy. A veces puede ser una información útil como usuaria, pero es principalmente útil para quienes ejecuten *guix publish* (véase Sección 7.10 [Invocación de *guix publish*], página 149).

Esta es una ejecución de ejemplo:

```
  $ guix weather --substitute-urls=https://guix.example.org
  computing 5,872 package derivations for x86_64-linux...
  looking for 6,128 store items on https://guix.example.org..
  updating list of substitutes from 'https://guix.example.org'... 100.0%
  https://guix.example.org
  43.4% substitutes available (2,658 out of 6,128)
  7,032.5 MiB of nars (compressed)
  19,824.2 MiB on disk (uncompressed)
  0.030 seconds per request (182.9 seconds in total)
  33.5 requests per second
  9.8% (342 out of 3,470) of the missing items are queued
  867 queued builds
  x86_64-linux: 518 (59.7%)
  i686-linux: 221 (25.5%)
  aarch64-linux: 128 (14.8%)
  build rate: 23.41 builds per hour
  x86_64-linux: 11.16 builds per hour
  i686-linux: 6.03 builds per hour
  aarch64-linux: 6.41 builds per hour
```

Como puede ver, informa de la fracción de todos los paquetes para los cuales hay sustituciones en el servidor—independientemente de que las sustituciones estén activadas, e
Capítulo 7: Utilidades

independientemente de si la clave de firma del servidor está autorizada. También informa del tamaño de los archivos comprimidos (“nar”) proporcionados por el servidor, el tamaño que los elementos correspondientes del almacén ocupan en el almacén (asumiendo que la deduplicación está apagada) y el caudal de proceso del servidor. La segunda parte proporciona estadísticas de integración continua (CI), si el servidor lo permite. Además, mediante el uso de la opción --coverage, guix weather puede enumerar sustituciones de paquetes “importantes” que no se encuentren en el servidor (véase más adelante).

Para conseguirlo, guix weather consulta los metadatos HTTP(S) (narinfo) de todos los elementos relevantes del almacén. Como guix challenge, ignora las firmas en esas sustituciones, lo cual es inocuo puesto que la orden únicamente obtiene estadísticas y no puede instalar esas sustituciones.

La sintaxis general es:

```
guix weather opciones... [paquetes...]```

When packages is omitted, guix weather checks the availability of substitutes for all the packages, or for those specified with --manifest; otherwise it only considers the specified packages. It is also possible to query specific system types with --system. guix weather exits with a non-zero code when the fraction of available substitutes is below 100%.

The available options are listed below.

```
--substitute-urls=urls

urls es la lista separada por espacios de URL de servidores de sustituciones a consultar. Cuando se omite esta opción, el conjunto predeterminado de servidores de sustituciones es el consultado.
```

```
--system=sistema
-s sistema

Consulta sustituciones para sistema—por ejemplo, aarch64-linux. Esta opción se puede repetir, en cuyo caso guix weather consultará las sustituciones para varios tipos de sistema.
```

```
--manifest=fichero

En vez de consultar las sustituciones de todos los paquetes, consulta únicamente los especificados en fichero. fichero debe contener un manifiesto, como el usado en la opción -m de guix package (véase Sección 4.2 [Invocación de guix package], página 36).

This option can be repeated several times, in which case the manifests are concatenated.
```

```
--coverage[=numero]
-c [numero]

Informa de la cobertura de sustituciones para paquetes: enumera paquetes con al menos número dependientes (cero por omisión) para los cuales no haya sustituciones disponibles. Los paquetes dependientes en sí no se enumeran: si b depende de a y a no tiene sustituciones disponibles, únicamente se muestra a, aunque b normalmente no tenga sustituciones tampoco. El resultado es más o menos así:

```
$ guix weather --substitute-urls=https://ci.guix.gnu.org -c 10

calculando 8.983 derivaciones de paquete para x86_64-linux...
```
buscando 9.343 elementos del almacén en https://ci.guix.gnu.org... |
actualizando sustituciones desde 'https://ci.guix.gnu.org'... 100.0% |
https://ci.guix.gnu.org
64,7% sustituciones disponibles (6.047 de 9.343)
...
Faltan 2502 paquetes de 'https://ci.guix.gnu.org' para 'x86_64-linux', entre los cuales:
58 kcoreaddons@5.49.0 /gnu/store/...-kcoreaddons-5.49.0
46 qgpgme@1.11.1 /gnu/store/...-qgpgme-1.11.1
37 perl-http-cookiejar@0.008 /gnu/store/...-perl-http-cookiejar-0.008
...
Lo que este ejemplo muestra es que kcoreaddons y presumiblemente los 58 paquetes que dependen de él no tienen sustituciones disponibles en ci.guix.info; del mismo modo que qgpgme y los 46 paquetes que dependen de él.
Si es una desarrolladora Guix, o si se encuentra a cargo de esta granja de construcción, probablemente quiera inspeccionar estos paquetes con más detalle: simplemente puede que su construcción falle.

--display-missing
Display the list of store items for which substitutes are missing.

7.15 Invocación de guix processes

La orden guix processes puede ser útil a desarrolladoras y administradoras de sistemas, especialmente en máquinas multiusuario y en granjas de construcción: enumera las sesiones actuales (conexiones al daemon), así como información sobre los procesos envueltos. A continuación puede verse un ejemplo de la información que devuelve:

```
$ sudo guix processes
SessionPID: 19002
ClientPID: 19090
ClientCommand: guix environment --ad-hoc python

SessionPID: 19402
ClientPID: 19367
ClientCommand: guix publish -u guix-publish -p 3000 -C 9 ...

SessionPID: 19444
ClientPID: 19419
ClientCommand: cuirass --cache-directory /var/cache/cuirass ...
LockHeld: /gnu/store/...-perl-ipc-cmd-0.96.lock
LockHeld: /gnu/store/...-python-six-bootstrap-1.11.0.lock
LockHeld: /gnu/store/...-libjpeg-turbo-2.0.0.lock
ChildProcess: 20495: guix offload x86_64-linux 7200 1 28800
ChildProcess: 27733: guix offload x86_64-linux 7200 1 28800
ChildProcess: 27793: guix offload x86_64-linux 7200 1 28800
```

6 Las sesiones remotas, cuando guix-daemon se ha iniciado con --listen especificando un punto de conexión TCP, no son enumeradas.
En este ejemplo vemos que guix-daemon tiene tres clientes: guix environment, guix publish y la herramienta de integración continua Cuirass; sus identificadores de proceso (PID) se muestran en el campo ClientPID. El campo SessionPID proporciona el PID del subproceso de guix-daemon de cada sesión en particular.

El campo LockHeld muestra qué elementos del almacén están bloqueados actualmente por cada sesión, lo que corresponde a elementos del almacén en construcción o sustitución (el campo LockHeld no se muestra cuando guix processes no se ejecutó como root). Por último, mediante el campo ChildProcess entendemos que esas tres construcciones están siendo delegadas (véase Sección 2.4.2 [Configuración de delegación del daemon], página 11).

La salida está en formato Recutils por lo que podemos usar la útil orden recsel para seleccionar sesiones de interés (véase Sección “Selection Expressions” en GNU recutils manual). Como un ejemplo, la siguiente orden muestra la línea de órdenes y el PID del cliente que inició la construcción de un paquete Perl:

```
$ sudo guix processes | \
   recsel -p ClientPID,ClientCommand -e 'LockHeld ~ "perl"'
ClientPID: 19419
ClientCommand: cuirass --cache-directory /var/cache/cuirass ...
```
8 Configuración del sistema

El sistema Guix permite un mecanismo de configuración del sistema completo consistente. Con esto queremos decir que todos los aspectos de la configuración global del sistema—como los servicios disponibles, la zona horaria y la configuración de localización, las cuentas de usuarios—se declaran en un lugar único. Dicha configuración del sistema puede ser instanciada—es decir, hecha efectiva.

Una de las ventajas de poner toda la configuración del sistema bajo el control de Guix es que permite actualizaciones transaccionales del sistema, y hace posible volver a una instancia previa del sistema, en caso de que haya algún problema con la nueva (véase Sección 4.1 [Características], página 35). Otra ventaja es que hace fácil replicar exactamente la misma configuración entre máquinas diferentes, o en diferentes momentos, sin tener que utilizar herramientas de administración adicionales sobre las propias herramientas del sistema.

Esta sección describe este mecanismo. Primero nos enfocaremos en el punto de vista de la administradora del sistema—explicando cómo se configura e instancia el sistema. Después mostraremos cómo puede extenderse este mecanismo, por ejemplo para añadir nuevos servicios del sistema.

8.1 Uso de la configuración del sistema

El sistema operativo se configura proporcionando una declaración operating-system en un fichero que pueda ser proporcionado a la orden guix system (véase Sección 8.14 [Invocación de guix system], página 399). Una configuración simple, con los servicios predeterminados del sistema, el núcleo Linux-Libre predeterminado, un disco de RAM inicial y un cargador de arranque puede ser como sigue:

```plaintext
;; This is an operating system configuration template
;; for a "bare bones" setup, with no X11 display server.

(use-modules (gnu))
(use-service-modules networking ssh)
(use-package-modules screen)

(operating-system
    (host-name "komputilo")
    (timezone "Europe/Berlin")
    (locale "en_US.utf8")

    ;; Boot in "legacy" BIOS mode, assuming /dev/sdX is the
    ;; target hard disk, and "my-root" is the label of the target
    ;; root file system.
    (bootloader (bootloader-configuration
        (bootloader grub-bootloader)
        (target "/dev/sdX")))

    (file-systems (cons (file-system
        (device (file-system-label "my-root"))))
```


(mount-point "/")
(type "ext4")
%base-file-systems)

;; This is where user accounts are specified. The "root"
;; account is implicit, and is initially created with the
;; empty password.
(users (cons (user-account
  (name "alice")
  (comment "Bob’s sister")
  (group "users")
  ;; Adding the account to the "wheel" group
  ;; makes it a sudoer. Adding it to "audio"
  ;; and "video" allows the user to play sound
  ;; and access the webcam.
  (supplementary-groups '("wheel" "audio" "video")))
%base-user-accounts))

;; Globally-installed packages.
(packages (cons screen %base-packages))

;; Add services to the baseline: a DHCP client and
;; an SSH server.
(services (append (list (service dhcp-client-service-type)
  (service openssh-service-type
    (openssh-configuration
      (port-number 2222))))
%base-services)))

Este ejemplo debería ser auto-descriptivo. Algunos de los campos definidos anteriormente, como host-name y bootloader, son necesarios. Otros como packages y services, pueden omitirse, en cuyo caso obtienen un valor por defecto.

Más adelante se muestran los efectos de algunos de los campos más importantes (véase Sección 8.2 [Referencia de operating-system], página 168, para detalles acerca de todos los campos disponibles), y cómo instanciar el sistema operativo usando guix system.

Cargador de arranque

El campo bootloader describe el método que será usado para arrancar su sistema. Las máquinas basadas en procesadores Intel pueden arrancar en el “obsoleto” modo BIOS, como en el ejemplo previo. No obstante, máquinas más recientes usan la Interfaz Unificada Extensible de Firmware (UEFI) para arrancar. En ese caso, el capo bootloader debe contener algo parecido a esto:

(bootloader-configuration
 (bootloader grub-efi-bootloader)
 (target "/boot/efi"))
Capítulo 8: Configuración del sistema

Véase Sección 8.13 [Configuración del gestor de arranque], página 396, para más información sobre las opciones de configuración disponibles.

Paquetes visibles globalmente

El campo `packages` enumera los paquetes que serán visibles globalmente en el sistema, para todas las cuentas de usuario—es decir, en la variable de entorno `PATH` de cada usuario—además de los perfiles por usuario (véase Sección 4.2 [Invocación de guix package], página 36). La variable `%base-packages` proporciona todas las herramientas esperadas para tareas básicas y de administración—incluyendo las utilidades básicas GNU, las herramientas de red GNU, el editor de texto ligero GNU Zile, `find`, `grep`, etc. El ejemplo previo se añade GNU Screen a estos, tomado del módulo `(gnu packages screen)` (véase Sección 6.1 [Módulos de paquetes], página 76). La sintaxis `(list package output)` puede usarse para añadir una salida específica de un paquete:

```
(use-modules (gnu packages))
(use-modules (gnu packages dns))

(operating-system
 ;; ...
 (packages (cons (list bind "utils")
                %base-packages)))
```

Referirse a los paquetes por nombre de variable, como antes a `bind`, tiene la ventaja de evitar ambigüedades; también permite que errores tipográficos y demás obtengan un diagnóstico directo como “variables sin definir”. La parte problemática es que se necesita conocer qué módulo define qué paquete, y aumentar adecuadamente la línea de `use-package-modules`. Para evitar esto, se puede usar el procedimiento `specification->package` del módulo `(gnu packages)`, que devuelve el mejor paquete para un nombre dado, o nombre y versión:

```
(use-modules (gnu packages))

(operating-system
 ;; ...
 (packages (append (map specification->package
                      '("tcpdump" "htop" "gnupg@2.0")
                     %base-packages)))
```

Servicios del sistema

El campo `services` enumera los servicios del sistema disponibles cuando el sistema arranque (véase Sección 8.8 [Servicios], página 183). La declaración `operating-system` previa especifica que, además de los servicios básicos, queremos que el daemon de shell seguro OpenSSH espere conexiones por el puerto 2222 (véase Sección 8.8.4 [Servicios de red], página 200). En su implementación, `openssh-service-type` prepara todo para que `sshd` se inicie con las opciones de la línea de órdenes adecuadas, posiblemente generando bajo demanda los ficheros de configuración necesarios (véase Sección 8.17 [Definición de servicios], página 412).

De manera ocasional, en vez de usar los servicios básicos tal y como vienen, puede querer personalizarlos. Para hacerlo, use `modify-services` (véase Sección 8.17.3 [Referencia de servicios], página 416) para modificar la lista.
Por ejemplo, supongamos que quiere modificar guix-daemon y Mingetty (el punto de acceso al sistema por consola) en la lista %base-services (véase Sección 8.8.1 [Servicios base], página 183). Para hacerlo, puede escribir lo siguiente en su declaración de sistema operativo:

```
(define %mis-servicios
  ;; Mi propia lista de servicios
  (modify-services %base-services
    (guix-service-type config =>
      (guix-configuration
       (inherit config)
       (use-substitutes? #f)
       (extra-options '("--gc-keep-derivations")))))
    (mingetty-service-type config =>
      (mingetty-configuration
       (inherit config)))))

(operating-system
  ;; ...
  (services %mis-servicios))
```

Esto modifica la configuración—es decir, los parámetros de los servicios—de la instancia guix-service-type, y de todas las instancias de mingetty-service-type en la lista %base-services. Observe cómo se consigue: primero, enlazamos la configuración actual al identificador config en el cuerpo, y entonces escribimos el cuerpo de manera que evalúe a la configuración deseada. En particular, fíjese como se usa inherit para crear una nueva configuración que tiene los mismos valores que la configuración antigua, pero con unas pocas modificaciones.

La configuración para un uso típico de “escritorio”, con una partición de raíz cifrada, el servidor gráfico X11, GNOME y Xfce (las usuarias pueden escoger cual de estos entornos de escritorio usarán en la pantalla de inicio de sesión pulsando F1), gestión de red, gestión de energía y más, podría ser así:

```
;; This is an operating system configuration template
;; for a "desktop" setup with GNOME and Xfce where the
;; root partition is encrypted with LUKS.

(use-modules (gnu) (gnu system nss))
(use-service-modules desktop xorg)
(use-package-modules certs gnome)

(operating-system
  (host-name "antelope")
  (timezone "Europe/Paris")
  (locale "en_US.utf8")

  ;; Choose US English keyboard layout. The "altgr-intl"
  ;; variant provides dead keys for accented characters.
  (keyboard-layout (keyboard-layout "us" "altgr-intl")))
```
Use the UEFI variant of GRUB with the EFI System.
Partition mounted on /boot/efi.

```
(bootloader (bootloader-configuration
  (bootloader grub-efi-bootloader)
  (target "/boot/efi")
  (keyboard-layout keyboard-layout)))
```

Specify a mapped device for the encrypted root partition.
The UUID is that returned by 'cryptsetup luksUUID'.
```
(mapped-devices
  (list (mapped-device
    (source (uuid "12345678-1234-1234-1234-123456789abc"))
    (target "my-root")
    (type luks-device-mapping))))
```

```
(file-systems (append
  (list (file-system
    (device (file-system-label "my-root"))
    (mount-point "/")
    (type "ext4")
    (dependencies mapped-devices))
    (file-system
      (device (uuid "1234-ABCD" 'fat))
      (mount-point "/boot/efi")
      (type "vfat")))

  %base-file-systems))
```

```
(users (cons (user-account
  (name "bob")
  (comment "Alice's brother")
  (group "users")
  (supplementary-groups '("wheel" "netdev" "audio" "video"))
    %base-user-accounts))
```

This is where we specify system-wide packages.
```
(packages (append (list
  ;; for HTTPS access
  nss-certs
  ;; for user mounts
  gvfs)

  %base-packages))
```

Add GNOME and Xfce---we can choose at the log-in screen
by clicking the gear. Use the "desktop" services, which
include the X11 log-in service, networking with
Capítulo 8: Configuración del sistema

;;; NetworkManager, and more.
(services (append (list 
    (service gnome-desktop-service-type) 
    (service xfce-desktop-service-type) 
    (set-xorg-configuration 
      (xorg-configuration 
        (keyboard-layout keyboard-layout)))) 
  %desktop-services))

;;; Allow resolution of '.local' host names with mDNS.
(name-service-switch %mdns-host-lookup-nss))

Un sistema gráfico con una selección de gestores de ventanas ligeros en vez de entornos de escritorio completos podría ser así:

;;; This is an operating system configuration template
;;; for a "desktop" setup without full-blown desktop
;;; environments.

(use-modules (gnu) (gnu system nss))
(use-service-modules desktop)
(use-package-modules bootloaders certs ratpoison suckless wm)

(operating-system 
  (host-name "antelope")
  (timezone "Europe/Paris")
  (locale "en_US.utf8")

;;; Use the UEFI variant of GRUB with the EFI System
;;; Partition mounted on /boot/efi.
(bootloader (bootloader-configuration 
    (bootloader grub-efi-bootloader) 
    (target "/boot/efi")))

;;; Assume the target root file system is labelled "my-root", 
;;; and the EFI System Partition has UUID 1234-ABCD.
(file-systems (append 
    (list (file-system 
        (device (file-system-label "my-root")) 
        (mount-point "/") 
        (type "ext4")) 
      (file-system 
        (device (uuid "1234-ABCD" 'fat)) 
        (mount-point "/boot/efi") 
        (type "vfat"))))

%base-file-systems)

(users (cons (user-account 
    (name "alice"))
(comment "Bob's sister")
(group "users")
(supplementary-groups '("wheel" "netdev" "audio" "video"))

;;; Add a bunch of window managers; we can choose one at
;;; the log-in screen with F1.
(packages (append (list
    ;; window managers
    ratpoison i3-wm i3status dmenu
    ;; for HTTPS access
    nss-certs)
%base-packages))

;;; Use the "desktop" services, which include the X11
;;; log-in service, networking with NetworkManager, and more.
(services %desktop-services)

;;; Allow resolution of ".local" host names with mDNS.
(name-service-switch %mdns-host-lookup-nss))

Este ejemplo se refiere al sistema de ficheros /boot/efi por su UUID 1234-ABCD. Substituya este UUID con el UUID correcto en su sistema, como el devuelto por la orden blkid.

Véase Sección 8.8.7 [Servicios de escritorio], página 237, para la lista exacta de servicios proporcionados por %desktop-services. Véase Sección 8.10 [Certificados X.509], página 390, para información sobre el paquete nss-certs usado aquí.

De nuevo, %desktop-services es simplemente una lista de objetos de servicios. Si desea borrar servicios de aquí, puede hacerlo usando procedimientos de filtrado de listas (véase Sección “SRFI-1 Filtering and Partitioning” en GNU Guile Reference Manual). Por ejemplo, la siguiente expresión devuelve una lista que contiene todos los servicios en %desktop-services excepto el servicio Avahi:

(remove (lambda (service)
    (eq? (service-kind service) avahi-service-type))
%desktop-services)

Instanciación del sistema

Asumiendo que la declaración de operating-system se encuentra en el fichero mi-configuración-del-sistema.scm, la orden guix system mi-conf-del-sistema.scm instancia esa configuración, y la convierte en la entrada predeterminada de GRUB en el arranque (véase Sección 8.14 [Invocación de guix system], página 399).

La manera habitual de cambiar la configuración del sistema es actualizar este fichero y volver a ejecutar guix system reconfigure. Nunca se deberían tocar los ficheros en /etc o ejecutar órdenes que modifiquen el estado del sistema como useradd o grub-install. De hecho, debe evitarlo ya que no únicamente anularía su garantía sino que también le impediría volver a una versión previa de su sistema, en caso de necesitarlo.
Hablando de vuelta atrás, cada vez que ejecuta `guix system reconfigure` se crea una nueva generación del sistema—sin modificar o borrar generaciones previas. Las generaciones previas tienen una entrada en el menú del cargador de arranque, lo que permite arrancarlas en caso de que algo funcionase mal en las últimas generaciones. Tranquilizador, ¿no? La orden `guix system list-generations` enumera las generaciones del sistema disponibles en el disco. Es también posible volver a una versión previa con las órdenes `guix system roll-back` y `guix system switch-generation`.

Aunque la orden `guix system reconfigure` no modificará las generaciones previas, debe tener cuidado cuando la generación actual no es la última (por ejemplo, después de invocar `guix system roll-back`), ya que la operación puede sobreescibir una generación posterior (véase Sección 8.14 [Invocación de guix system], página 399).

### La interfaz programática

A nivel Scheme, el grueso de una declaración `operating-system` se instancia con el siguiente procedimiento monádico (véase Sección 6.6 [La mónada del almacén], página 104):

```scheme
operating-system-derivation so

[Procedimiento monádico]

Devuelve una derivación que construye `so`, un objeto `operating-system` (véase Sección 6.5 [Derivaciones], página 101).

La salida de la derivación es un único directorio que hace referencia a todos los paquetes, ficheros de configuración y otros ficheros auxiliares necesarios para instanciar `so`.

Este procedimiento se proporciona por el módulo `(gnu system)`. Junto con `(gnu services)` (véase Sección 8.8 [Servicios], página 183), este módulo contiene los entresijos del sistema Guix. ¡Asegúrese de echarle un vistazo!

### 8.2 Referencia de `operating-system`

Esta sección resume todas las opciones disponibles en las declaraciones de `operating-system` (véase Sección 8.1 [Uso de la configuración del sistema], página 161).

```scheme
operating-system

[Tipo de datos]

Este es el tipo de datos que representa la configuración del sistema operativo. Con ello queremos decir toda la configuración global del sistema, no la configuración específica de las usuarias (véase Sección 8.1 [Uso de la configuración del sistema], página 161).

`kernel` (predeterminado: `linux-libre`)

El objeto del paquete del núcleo del sistema operativo usado¹.

`kernel-loadable-modules` (default: '())

A list of objects (usually packages) to collect loadable kernel modules from—e.g. `(list ddcci-driver-linux)`.

`kernel-arguments` (predeterminados: '("quiet"))

Lista de cadenas o expresiones-G que representan parámetros adicionales a pasar en la línea de órdenes del núcleo—por ejemplo, "("console=ttys0")".

¹ Actualmente únicamente está disponible el núcleo Linux-libre. En el futuro será posible usar GNU Hurd.
**bootloader**
El objeto de configuración del cargador de arranque del sistema. Véase Sección 8.13 [Configuración del gestor de arranque], página 396.

**label**
Es una etiqueta (una cadena) con la que aparecerá en el menú del cargador de arranque. La etiqueta predeterminada incluye el nombre y la versión del núcleo.

**keyboard-layout** (predeterminada: #f)
Este campo especifica la distribución de teclado usada para la consola. Puede ser o bien #f, en cuyo caso se usa la distribución predeterminada (normalmente Inglés de EEUU), o un registro <keyboard-layout>. Esta distribución de teclado se hace efectiva tan pronto el núcleo haya arrancado. Por ejemplo, la distribución de teclado está en efecto cuando introduzca una contraseña si su sistema de ficheros raíz se encuentra en un dispositivo traducido luks-device-mapping (véase Sección 8.4 [Dispositivos traducidos], página 175).

Nota: Esto no especifica la distribución de teclado usada por el cargador de arranque, ni tampoco la usada por el servidor gráfico. Véase Sección 8.13 [Configuración del gestor de arranque], página 396, para información sobre cómo especificar la distribución de teclado del cargador de arranque. Véase Sección 8.8.5 [Sistema X Window], página 218, para información sobre cómo especificar la distribución de teclado usada por el sistema de ventanas X.

**initrd-modules** (predeterminados: %base-initrd-modules)
La lista de módulos del núcleo Linux que deben estar disponibles en el disco inicial de RAM. Véase Sección 8.12 [Disco en RAM inicial], página 393.

**initrd** (predeterminado: base-initrd)
Un procedimiento que devuelve un disco inicial de RAM para el núcleo Linux. Este campo se proporciona para permitir personalizaciones de bajo nivel y no debería ser necesario para un uso habitual. Véase Sección 8.12 [Disco en RAM inicial], página 393.

**firmware** (predeterminado: %base-firmware)
Lista de paquetes de firmware que pueden ser cargados por el núcleo del sistema operativo.

El valor predeterminado incluye el firmware necesario para dispositivos WiFi basados en Atheros y Broadcom (módulos Linux-libre ath9k y b43-open, respectivamente). Véase Sección 3.2 [Consideraciones sobre el hardware], página 23, para más información sobre hardware soportado.

**host-name**
El nombre de la máquina.

**hosts-file**
Un objeto tipo-fichero (véase Sección 6.7 [Expresiones-G], página 108) para ser usado como /etc/hosts (véase Sección “Host Names” en The
El predeterminado es un fichero con entradas para *localhost* y *host-name*.

**mapped-devices** (predeterminados: ‘()’)
Una lista de dispositivos traducidos. Véase Sección 8.4 [Dispositivos traducidos], página 175.

**file-systems**
Una lista de sistemas de ficheros. Véase Sección 8.3 [Sistemas de ficheros], página 172.

**swap-devices** (predeterminados: ‘()’)
Una lista de cadenas que identifiquen dispositivos o ficheros usados como “espacio de intercambio” (véase Sección “Memory Concepts” en *The GNU C Library Reference Manual*). Por ejemplo ‘("/dev/sda3")’ o ‘("/fichero-intercambio")’. Es posible especificar un fichero de intercambio en un sistema de ficheros en un dispositivo traducido, siempre que la traducción y el sistema de ficheros se especifiquen también. Véase Sección 8.4 [Dispositivos traducidos], página 175, y Sección 8.3 [Sistemas de ficheros], página 172.

**users** (predeterminadas: %base-user-accounts)
**groups** (predeterminados: %base-groups)
Lista de cuentas de usuaria y grupos. Véase Sección 8.5 [Cuentas de usuaria], página 176.

Si la lista de *usuarios* carece de una cuenta de usuaria con UID 0, una cuenta “root” con UID 0 se añade automáticamente.

**skeletons** (predeterminados: (default-skeletons))
Una lista de tuplas de nombre de fichero de destino/objeto tipo-fichero (véase Sección 6.7 [Expresiones-G], página 108). Estos son los ficheros de esqueleto que se añadirán al directorio de las cuentas de usuaria que se cree.

Por ejemplo, un valor válido puede parecer algo así:
```
'((".bashrc",(plain-file "bashrc" "echo Hola\n"))
 (".guile",(plain-file "guile"
  
    "(use-modules (ice-9 readline))
     (activate-readline)")))
```

**issue** (predeterminado: %default-issue)
Una cadena que denota el contenido del fichero /etc/issue, que se muestra cuando las usuarias ingresan al sistema en una consola de texto.

**packages** (predeterminados: %base-packages)
El conjunto de paquetes instalados en el perfil global, que es accesible en /run/current-system/profile.

El conjunto predeterminado incluye utilidades básicas y es una buena práctica instalar utilidades no-básicas en los perfiles de las usuarias (véase Sección 4.2 [Invocación de guix package], página 36).
timezone Una cadena que identifica la zona horaria—por ejemplo, "Europe/Paris".
   Puede ejecutar la orden tzselect para encontrar qué cadena de zona horaria corresponde con su región. Elegir una zona horaria no válida provoca un fallo en guix system.

locale (predeterminado: "en_US.utf8")

locale-definitions (predeterminadas: %default-locale-definitions)
   La lista de definiciones de localizaciones a compilar y que puede ser usada en tiempo de ejecución. Véase Sección 8.7 [Localizaciones], página 181.

locale-libcs (predeterminadas: (list glibc))
   La lista de paquetes GNU libc cuyos datos de localización y herramientas son usadas para las definiciones de localizaciones. Véase Sección 8.7 [Localizaciones], página 181, para consideraciones de compatibilidad que justifican esta opción.

name-service-switch (predeterminado: %default-nss)
   Configuración del selector de servicios de nombres de libc (NSS)—un objeto <name-service-switch>. Véase Sección 8.11 [Selector de servicios de nombres], página 391, para detalles.

services (predeterminados: %base-services)
   Una lista de objetos service denotando los servicios del sistema. Véase Sección 8.8 [Servicios], página 183.

essential-services (predeterminados: ...)
   La lista de “servicios esenciales”—es decir, cosas como instancias de system-service-type y host-name-service-type (véase Sección 8.17.3 [Referencia de servicios], página 416), las cuales se derivan de su definición de sistema operativo en sí. Como usuaria nunca debería modificar este campo.

pam-services (predeterminados: (base-pam-services))
   Servicios de los módulos de identificación conectables (PAM) de Linux.

setuid-programs (predeterminados: %setuid-programs)
   Lista de expresiones-G con valores de cadena que denotan los programas setuid. Véase Sección 8.9 [Programas con setuid], página 389.

sudoers-file (predeterminado: %sudoers-specification)
   El contenido de /etc/sudoers como un objeto tipo-fichero (véase Sección 6.7 [Expresiones-G], página 108).
   Este fichero especifica qué usuarias pueden usar la orden sudo, lo que se les permite hacer y qué privilegios pueden obtener. El comportamiento predefinido es que únicamente root y los miembros del grupo wheel pueden usar sudo.
this-operating-system [Tipo de datos]
Cuando se usa en el ámbito léxico de un campo de una definición de sistema operativo, este identificador está enlazado al sistema operativo en definición.
El siguiente ejemplo muestra cómo hacer referencia al sistema operativo en definición en la definición del campo label:

  (use-modules (gnu) (guix))

  (operating-system
    ;; ...
    (label (package-full-name
            (operating-system-kernel this-operating-system))))

Es un error hacer referencia a this-operating-system fuera de una definición de sistema operativo.

8.3 Sistemas de ficheros
La lista de sistemas de ficheros que deben montarse se especifica en el campo file-systems de la declaración del sistema operativo (véase Sección 8.1 [Uso de la configuración del sistema], página 161). Cada sistema de ficheros se declara usando la forma file-system, como en el siguiente ejemplo:

  (file-system
    (mount-point "/home")
    (device "/dev/sda3")
    (type "ext4"))

Como es habitual, algunos de los campos son obligatorios—aquellos mostrados en el ejemplo previo—mientras que otros pueden omitirse. Se describen a continuación.

file-system [Tipo de datos]
Objetos de este tipo representan los sistemas de ficheros a montar. Contienen los siguientes campos:

type Este campo es una cadena que especifica el tipo de sistema de ficheros—por ejemplo, "ext4".

mount-point Designa la ruta donde el sistema de ficheros debe montarse.

device Nombra la “fuente” del sistema de ficheros. Puede ser una de estas tres opciones: una etiqueta de sistema de ficheros, un UUID de sistema de ficheros o el nombre de un nodo /dev. Las etiquetas y UUID ofrecen una forma de hacer referencia a sistemas de ficheros sin codificar su nombre de dispositivo actual\(^2\).

Las etiquetas del sistema de ficheros se crean mediante el uso del procedimiento file-system-label, los UUID se crean mediante el uso de uuid

---

\(^2\) Fíjese que, aunque es tentador usar /dev/disk/by-uuid y nombres de dispositivo similares para obtener el mismo resultado, no es lo recomendado: estos nodo especiales de dispositivos se crean por el daemon udev y puede no estar disponible cuando el dispositivo sea montado.
y los nodos /dev son simples cadenas. A continuación se proporciona un ejemplo de un sistema de ficheros al que se hace referencia mediante su etiqueta, como es mostrada por la orden e2label:

```
(file-system
 (mount-point "/home")
 (type "ext4")
 (device (file-system-label "mi-home")))
```

Los UUID se convierten desde su representación en forma de cadena (como se muestra con la orden tune2fs -l) mediante el uso de la forma uuid3, como sigue:

```
(file-system
 (mount-point "/home")
 (type "ext4")
 (device (uuid "4dab5feb-d176-45de-b287-9b0a6e4c01cb")))
```

Cuando la fuente de un sistema de ficheros es un dispositivo traducido (véase Sección 8.4 [Dispositivos traducidos], página 175), su campo device debe hacer referencia al nombre del dispositivo traducido—por ejemplo, "/dev/mapper/particion-raiz". Esto es necesario para que el sistema sepa que el montaje del sistema de ficheros depende del establecimiento de la traducción de dispositivos correspondiente.

**flags** (predeterminadas: '() )

Es una lista de símbolos que indican opciones del montado. Las opciones reconocidas incluyen read-only4, bind-mount5, no-dev (prohibición del acceso a ficheros especiales), no-suid (ignora los bits setuid y setgid), no-atime (no actualiza la marca de tiempo del acceso a ficheros), strict-atime (actualiza la marca de tiempo del acceso a ficheros), lazy-time (únicamente actualiza la marca de tiempo en la versión en memoria del nodo-i) y no-exec (no permite de la ejecución de programas). Véase Sección “Mount-Unmount-Remount” en *The GNU C Library Reference Manual*, para más información sobre estas opciones.

**options** (predeterminadas: #f)

Es o bien #f, o bien una cadena que denota las opciones de montaje proporcionadas al controlador del sistema de ficheros. Véase Sección “Mount-Unmount-Remount” en *The GNU C Library Reference Manual* para obtener detalles, y ejecute man 8 mount para conocer las opciones de varios sistemas de ficheros.

**mount?** (predeterminado: #t)

Este valor indica si debe montarse el sistema de ficheros automáticamente al iniciar el sistema. Cuando se establece como #f, el sistema de ficheros

---

3 La forma uuid espera un UUID de 16 bytes como se define en la RFC 4122 (https://tools.ietf.org/html/rfc4122). Este es el formato de UUID que usan la familia de sistemas de ficheros ext2 y otros, pero es diferente de los “UUID” de los sistemas de ficheros FAT, por ejemplo.  
4 NdT: modo de sólo lectura.  
5 NdT: montaje enlazado.
tiene una entrada en `/etc/fstab` (el cual es leído por la orden `mount`) pero no se montará automáticamente.

**needed-for-boot?** (predeterminado: `#f`)
Este valor lógico indica si el sistema de ficheros es necesario para el arranque. Si es verdadero, el sistema de ficheros se monta al cargar el disco inicial de RAM (initrd). Este es siempre el caso, por ejemplo, para el sistema de ficheros raíz.

**check?** (predeterminado: `#t`)
Este valor lógico indica si el sistema de ficheros se debe comprobar en busca de errores antes de montarse.

**create-mount-point?** (predeterminado: `#f`)
Cuando es verdadero, el punto de montaje es creado si no existía previamente.

**dependencies** (predeterminadas: `'(())`)
Una lista de objetos `<file-system>` o `<mapped-device>` que representan sistemas de ficheros que deben montarse o dispositivos traducidos que se deben abrir antes (y desmontar o cerrar después) que el declarado.

Como ejemplo, considere la siguiente jerarquía de montajes: /
`/sys/fs/cgroup` es una dependencia de `/sys/fs/cgroup/cpu` y `/sys/fs/cgroup/memory`

Otro ejemplo es un sistema de ficheros que depende de un dispositivo traducido, por ejemplo una partición cifrada (véase Sección 8.4 [Dispositivos traducidos], página 175).

El módulo (`gnu system file-systems`) exporta las siguientes variables útiles.

**%base-file-systems** [Variable Scheme]
Estos son los sistemas de ficheros esenciales que se necesitan en sistemas normales, como `%pseudo-terminal-file-system` y `%immutable-store` (véase a continuación). Las declaraciones de sistemas operativos deben contener siempre estos al menos.

**%pseudo-terminal-file-systems** [Variable Scheme]
El sistema de ficheros que debe montarse como `/dev/pts`. Permite la creación de pseudoterminales a través de `openpty` y funciones similares (véase Sección “Pseudoterminals” en The GNU C Library Reference Manual). Los pseudoterminales son usados por emuladores de terminales como `xterm`.

**%shared-memory-file-system** [Variable Scheme]
Este sistema de ficheros se monta como `/dev/shm` y se usa para permitir el uso de memoria compartida entre procesos (véase Sección “Memory-mapped I/O” en The GNU C Library Reference Manual).

**%immutable-store** [Variable Scheme]
Este sistema de ficheros crea un montaje enlazado (“bind-mount”) de `/gnu/store`, permitiendo solo el acceso de lectura para todas las usuarias incluyendo a `root`. Esto previene modificaciones accidentales por software que se ejecuta como `root` o por las administradoras del sistema.
El daemon sí es capaz de escribir en el almacén: vuelve a montar /gnu/store en modo lectura-escritura en su propio “espacio de nombres”.

%binary-format-file-system
El sistema de ficheros binfmt_misc, que permite que el manejo de tipos de ficheros ejecutables arbitrarios se delegue al espacio de usuaria. Necesita la carga del módulo del núcleo binfmt.ko.

%fuse-control-file-system
El sistema de ficheros fusectl, que permite a usuarias sin privilegios montar y desmontar sistemas de ficheros de espacio de usuaria FUSE. Necesita la carga del módulo del núcleo fuse.ko.

8.4 Dispositivos traducidos
El núcleo Linux tiene una noción de traducción de dispositivos: un dispositivo de bloques, como una partición de disco duro, puede traducirse en otro dispositivo, habitualmente en /dev/mapper/, con un procesamiento adicional sobre los datos que fluyen a través de ella. Un ejemplo típico es la traducción de dispositivos para el cifrado: todas las escrituras en el dispositivo traducido se cifran, y todas las lecturas se descifran, de forma transparente. Guix extiende esta noción considerando cualquier dispositivo o conjunto de dispositivos que son transformados de alguna manera para crear un nuevo dispositivo; por ejemplo, los dispositivos RAID se obtienen ensamblando otros dispositivos, como discos duros o particiones, en uno nuevo que se comporta como una partición. Otros ejemplos, todavía no implementados, son los volúmenes lógicos LVM.

Los dispositivos traducidos se declaran mediante el uso de la forma mapped-device, definida a continuación; ejemplos más adelante.

mapped-device
Objetos de este tipo representan traducciones de dispositivo que se llevarán a cabo cuando el sistema arranque.

source
Puede ser tanto una cadena que especifica el nombre de un dispositivo de bloques a traducir, como "/dev/sda3", o una lista de dichas cadenas cuando varios dispositivos necesitan ser ensamblados para crear uno nuevo.

target
Esta cadena especifica el nombre del dispositivo traducido resultante. Para traductores del núcleo como dispositivos de cifrado del tipo luks-device-mapping, especificar "mi-particion" produce la creación del dispositivo "/dev/mapper/mi-particion". Para dispositivos RAID de tipo raid-device-mapping, el nombre del dispositivo completo como "/dev/md0" debe ser proporcionado.

type
Debe ser un objeto mapped-device-kind, que especifica cómo source se traduce a target.

---

6 Fíjese que GNU Hurd no diferencia entre el concepto de un “dispositivo traducido” y el de un sistema de ficheros: ambos se reducen a traducir operaciones de entrada/salida realizadas en un fichero a operaciones en su almacenamiento subyacente. Por tanto, Hurd implementa dispositivos traducidos, como sistemas de ficheros, usando el mecanismo genérico de traducción (véase Sección “Translators” en The GNU Hurd Reference Manual).
luks-device-mapping [Variable Scheme]
Define el cifrado de bloques LUKS mediante el uso de la orden cryptsetup del paquete del mismo nombre. Depende del módulo dm-crypt del núcleo Linux.

raid-device-mapping [Variable Scheme]
Define un dispositivo RAID, el cual se ensambla mediante el uso de la orden mdadm del paquete del mismo nombre. Requiere la carga del módulo del núcleo Linux para el nivel RAID apropiado, como raid456 para RAID-4, RAID-5 o RAID-6, o raid10 para RAID-10.

El siguiente ejemplo especifica una traducción de /dev/sda3 a /dev/mapper/home mediante el uso de LUKS—la configuración de claves unificada de Linux (https://gitlab.com/cryptsetup/cryptsetup), un mecanismo estándar para cifrado de disco. El dispositivo /dev/mapper/home puede usarse entonces como el campo device de una declaración file-system (véase Sección 8.3 [Sistemas de ficheros], página 172).

(mapped-device
  (source "/dev/sda3")
  (target "home")
  (type luks-device-mapping))

De manera alternativa, para independizarse de la numeración de dispositivos, puede obtenerse el UUID LUKS (identificador único) del dispositivo fuente con una orden así:
cryptsetup luksUUID /dev/sda3
y usarlo como sigue:
(mapped-device
  (source (uuid "cb67fc72-0d54-4c88-9d4b-b225f30b0f44"))
  (target "home")
  (type luks-device-mapping))

También es deseable cifrar el espacio de intercambio, puesto que el espacio de intercambio puede contener información sensible. Una forma de conseguirlo es usar un fichero de intercambio en un sistema de ficheros en un dispositivo traducido a través del cifrado LUKS. Véase Sección 3.4 [Particionado del disco], página 25, para un ejemplo.

Un dispositivo RAID formado por las particiones /dev/sda1 y /dev/sdb1 puede declararse como se muestra a continuación:
(mapped-device
  (source (list "/dev/sda1" "/dev/sdb1"))
  (target "/dev/md0")
  (type raid-device-mapping))

El dispositivo /dev/md0 puede usarse entonces como el campo device de una declaración file-system (véase Sección 8.3 [Sistemas de ficheros], página 172). Fíjese que no necesita proporcionar el nivel RAID; se selecciona durante la creación inicial y formato del dispositivo RAID y después se determina automáticamente.

8.5 Cuentas de usuario
Los grupos y cuentas de usuario se gestionan completamente a través de la declaración operating-system. Se especifican con las formas user-account y user-group:
(user-account
Durante el arranque o tras la finalización de guix system reconfigure, el sistema se asegura de que únicamente las cuentas de usuaria y grupos especificados en la declaración operating-system existen, y con las propiedades especificadas. Por tanto, la creación o modificación de cuentas o grupos realizadas directamente invocando órdenes como useradd se pierden al reconfigurar o reiniciar el sistema. Esto asegura que el sistema permanece exactamente como se declaró.

user-account [Tipo de datos]
Objetos de este tipo representan cuentas de usuaria. Los siguientes miembros pueden ser especificados:

name El nombre de la cuenta de usuaria.
group Este es el nombre (una cadena) o identificador (un número) del grupo de usuarias al que esta cuenta pertenece.
supplementary-groups (predeterminados: '() )
Opcionalmente, esto puede definirse como una lista de nombres de grupo a los que esta cuenta pertenece.
uid (predeterminado: #f)
Este es el ID de usuaria para esta cuenta (un número), o #f. En el último caso, un número es seleccionado automáticamente por el sistema cuando la cuenta es creada.
comment (predeterminado: "")
Un comentario sobre la cuenta, como el nombre completo de la propietaria.
home-directory
Este es el nombre del directorio de usuaria de la cuenta.
create-home-directory? (predeterminado: #t)
Indica si el directorio de usuaria de esta cuenta debe ser creado si no existe todavía.
shell (predeterminado: Bash)
 Esto es una expresión-G denotando el nombre de fichero de un programa que será usado como shell (véase Sección 6.7 [Expresiones-G], página 108).
system? (predeterminado: #f)
Este valor lógico indica si la cuenta es una cuenta “del sistema”. Las cuentas del sistema se tratan a veces de forma especial; por ejemplo, los gestores gráficos de inicio no las enumeran.
password (predeterminada: \#f)

Normalmente deberías dejar este campo a \#f, inicializar la contraseña de usuario como root con la orden passwd, y entonces dejar a las usuarias cambiarla con passwd. Las contraseñas establecidas con passwd son, por supuesto, preservadas entre reinicio y reinicio, y entre reconfiguraciones.

Si usted realmente quiere tener una contraseña prefijada para una cuenta, entonces este campo debe contener la contraseña cifrada, como una cadena. Puedes usar el procedimiento crypt para este fin:

```
      (user-account
       (name "carlos")
       (group "users")

      ;; Especifica una contraseña inicial mediante un hash SHA-512.
      (password (crypt "ContraseñaInicial!" ";$6$abc")))
```

Nota: El hash de esta contraseña inicial estará disponible en un fichero en /gnu/store, legible por todas las usuarias, por lo que este método debe usarse con precaución.


Las declaraciones de grupos incluso son más simples:

```
      (user-group (name "estudiantes"))
```

user-group [Tipo de datos]

Este tipo es para grupos de usuarias. Hay únicamente unos pocos campos:

name El nombre del grupo.

id (predeterminado: \#f)

El identificador del grupo (un número). Si es \#f, un nuevo número es reservado automáticamente cuando se crea el grupo.

system? (predeterminado: \#f)

Este valor booleano indica si el grupo es un grupo “del sistema”. Los grupos del sistema tienen identificadores numéricos bajos.

password (predeterminada: \#f)

¿Qué? ¿Los grupos de usuarias pueden tener una contraseña? Bueno, aparentemente sí. A menos que sea \#f, este campo especifica la contraseña del grupo.

Por conveniencia, una variable contiene una lista con todos los grupos de usuarias básicos que se puede esperar:

```
%base-groups [Variable Scheme]

Esta es la lista de grupos de usuarias básicos que las usuarias y/o los paquetes esperan que estén presentes en el sistema. Esto incluye grupos como “root”, “wheel” y “users”, así como grupos usados para controlar el acceso a dispositivos específicos como “audio”, “disk” y “cdrom”.
```
Esta es la lista de cuentas de usuario básicas que los programas pueden esperar encontrar en un sistema GNU/Linux, como la cuenta “nobody”.

Fíjese que la cuenta de “root” no se incluye aquí. Es un caso especial y se añade automáticamente esté o no especificada.

8.6 Distribución de teclado

Para especificar qué hace cada tecla de su teclado, necesita decirle al sistema operativo qué distribución de teclado desea usar. La predeterminada, cuando no se especifica ninguna, es la distribución QWERTY de 105 teclas para PC de teclado inglés estadounidense. No obstante, las personas germano-parlantes habitualmente prefieren la distribución QWERTZ alemana, las franco-parlantes desearán la distribución AZERTY, etcétera; las hackers pueden preferir Dvorak o bépo, y pueden incluso desear personalizar más aún el efecto de determinadas teclas. Esta sección explica cómo hacerlo.

Hay tres componentes que desean conocer la distribución de su teclado:

- El cargador de arranque puede desear conocer cual es la distribución de teclado que desea usar (véase Sección 8.13 [Configuración del gestor de arranque], página 396). Esto es útil si desea, por ejemplo, asegurarse de que puede introducir la contraseña de cifrado de su partición raíz usando la distribución correcta.
- El núcleo del sistema operativo, Linux, la necesitará de manera que la consola se configure de manera adecuada (véase Sección 8.2 [Referencia de operating-system], página 168).
- El servidor gráfico, habitualmente Xorg, también tiene su propia idea de distribución de teclado (véase Sección 8.8.5 [Sistema X Window], página 218).

Guix le permite configurar las tres distribuciones por separado pero, afortunadamente, también le permite compartir la misma distribución de teclado para los tres componentes.

Las distribuciones de teclado se representan mediante registros creados con el procedimiento keyboard-layout de (gnu system keyboard). A imagen de la extensión de teclado de X (XKB), cada distribución tiene cuatro atributos: un nombre (habitualmente un código de idioma como “fi” para finés o “jp” para japonés), un nombre opcional de variante, un nombre opcional de modelo de teclado y una lista posiblemente vacía de opciones adicionales. En la mayor parte de los casos el nombre de la distribución es lo único que le interesará. Aquí tiene algunos ejemplos:

```guile
;; La distribución QWERTZ alemana. Se asume un modelo de
;; teclado "pc105" estándar.
(keyboard-layout "de")

;; La variante bépo de la distribución francesa.
(keyboard-layout "fr" "bepo")

;; La distribución catalana.
(keyboard-layout "es" "cat")

;; La distribución de teclado de latinoamérica. Además,
```
Capítulo 8: Configuración del sistema

;;; la tecla "Bloq Mayúš" se usa como una tecla "Ctrl"
;;; adicional, y la tecla "Menú" se usa como una tecla
;;; "Componer/Compose" para introducir letras acentuadas.
(keyboard-layout "latam"
   #:options '("ctrl:nocaps" "compose:menu"))

;;; La distribución rusa para un teclado ThinkPad.
(keyboard-layout "ru" #:model "thinkpad")

;;; La distribución estadounidense internacional, la cual es
;;; la distribución estadounidense junto a teclas muertas para
;;; introducir caracteres acentuados. Esta es para un teclado
;;; Apple MackBook.
(keyboard-layout "us" "intl" #:model "macbook78")

Véase el directorio share/X11/xkb del paquete xkeyboard-config para una lista completa de implementaciones de distribuciones, variantes y modelos.

Digamos que desea que su sistema use la distribución de teclado turca a lo largo de todo su sistema—cargador de arranque, consola y Xorg. Así es como sería su configuración del sistema:

;;; Usando la distribución turca para el cargador de
;;; arranque, la consola y Xorg.

(operating-system
 ;;; ...
 (keyboard-layout (keyboard-layout "tr")) ;para la consola
 (bootloader (bootloader-configuration
                (bootloader grub-efi-bootloader)
                (target "/boot/efi")
                (keyboard-layout keyboard-layout))) ;para GRUB
 (services (cons (set-xorg-configuration
                  (xorg-configuration
                   ;para Xorg
                   (keyboard-layout keyboard-layout))
                  %desktop-services)))

En el ejemplo previo, para GRUB y para Xorg, simplemente hemos hecho referencia al campo keyboard-layout definido previamente, pero también podíamos haber hecho referencia a una distribución diferente. El procedimiento set-xorg-configuration comunica la configuración de Xorg deseada al gestor gráfico de ingreso en el sistema, GDM por omisión.

Hemos tratado cómo especificar la distribución predeterminada del teclado de su sistema cuando arranca, pero también la puede modificar en tiempo de ejecución:

- Si usa GNOME, su panel de configuración tiene una entrada de “Región e Idioma” donde puede seleccionar una o más distribuciones de teclado.
- En Xorg, la orden setxkbmap (del paquete con el mismo nombre) le permite cambiar la distribución en uso actualmente. Por ejemplo, así es como cambiaría a la distribución Dvorak estadounidense:

  setxkbmap us dvorak
La orden `loadkeys` cambia la distribución de teclado en efecto en la consola Linux. No obstante, tenga en cuenta que `loadkeys` no usa la categorización de distribuciones de XKB descrita previamente. La orden a continuación carga la distribución francesa bépo:

```
loadkeys fr-bepo
```

### 8.7 Localizaciones

Una localización define convenciones culturales para una lengua y región del mundo particular (véase Sección “Localizaciones” en *The GNU C Library Reference Manual*). Cada localización tiene un nombre que típicamente tiene la forma de `lengua_territorio.codificación`—por ejemplo, `fr_LU.utf8` designa la localización para la lengua francesa, con las convenciones culturales de Luxemburgo, usando la codificación UTF-8.

Normalmente deseará especificar la localización predeterminada para la máquina usando el campo `locale` de la declaración `operating-system` (véase Sección 8.2 [Referencia de `operating-system`], página 168).

La localización seleccionada es automáticamente añadida a las definiciones de localización conocidas en el sistema si es necesario, con su codificación inferida de su nombre—por ejemplo, se asume que `bo_CN.utf8` usa la codificación UTF-8. Definiciones de localización adicionales pueden ser especificadas en el campo `locale-definitions` de `operating-system`—esto es útil, por ejemplo, si la codificación no puede ser inferida del nombre de la localización. El conjunto predeterminado de definiciones de localización incluye algunas localizaciones ampliamente usadas, pero no todas las disponibles, para ahorrar espacio.

Por ejemplo, para añadir la localización del frisio del norte para Alemania, el valor de dicho campo puede ser:

```
(cons (locale-definition
    (name "fy_DE.utf8") (source "fy_DE"))
  %default-locale-definitions)
```

De mismo modo, para ahorrar espacio, se puede desear que `locale-definitions` contenga únicamente las localizaciones que son realmente usadas, como en:

```
(list (locale-definition
    (name "ja_JP.eucjp") (source "ja_JP")
    (charset "EUC-JP")))
```

Las definiciones de localización compiladas están disponibles en `/run/current-system/locale/X.Y` donde `X.Y` es la versión de libc, que es la ruta donde la GNU libc contenida en Guix buscará los datos de localización. Esto puede ser sobrescrito usando la variable de entorno `LOCPATH` (véase [locales-and-locpath], página 19).

La forma `locale-definition` es proporcionada por el módulo (`gnu system locale`). Los detalles se proporcionan a continuación.

```
locale-definition  [Tipo de datos]
```

Este es el tipo de datos de una definición de localización.

source El nombre de la fuente para dicha localización. Habitualmente es la parte idioma_territorio del nombre de localización.

charset (predeterminado: "UTF-8") La “codificación de caracteres” o “conjunto de caracteres” para dicha localización, como lo define IANA (https://www.iana.org/assignments/character-sets).

%default-locale-definitions [Variable Scheme] Una lista de localizaciones UTF-8 usadas de forma común, usada como valor predeterminado del campo locale-definitions en las declaraciones operating-system. Estas definiciones de localizaciones usan la codificación normalizada para el fragmento tras el punto en el nombre (véase Sección “Using gettextized software” en The GNU C Library Reference Manual). Por lo que por ejemplo es válido uk_UA.utf8 pero no, digamos, uk_UA.UTF-8.

8.7.1 Consideraciones sobre la compatibilidad de datos de localización

Las declaraciones operating-system proporcionan un campo locale-libcs para especificar los paquetes GNU libc que se usará para compilar las declaraciones de localizaciones (véase Sección 8.2 [Referencia de operating-system], página 168). “¿Por qué debo preocuparme?”, puede preguntarse. Bueno, sucede que el formato binario de los datos de localización es ocasionalmente incompatible de una versión de libc a otra. Por ejemplo, un programa enlazado con la versión 2.21 de libc no puede leer datos de localización producidos con libc 2.22; peor aún, ese programa aborta en vez de simplemente ignorar los datos de localización incompatibles7. De manera similar, un programa enlazado con libc 2.22 puede leer la mayor parte, pero no todo, de los datos de localización de libc 2.21 (específicamente, los datos LC_COLLATE son incompatibles); por tanto las llamadas a setlocale pueden fallar, pero los programas no abortarán.

El “problema” con Guix es que las usuarias tienen mucha libertad: pueden elegir cuando e incluso si actualizar el software en sus perfiles, y pueden estar usando una versión de libc diferente de la que la administradora del sistema usó para construir los datos de localización comunes a todo el sistema.

Por suerte, las usuarias sin privilegios también pueden instalar sus propios datos de localización y definir GUIX_LOCPATH adecuadamente (véase locales-and-locpath], página 19).

No obstante, es mejor si los datos de localización globales del sistema en /run/current-system/locale se construyen para todas las versiones de libc realmente en uso en el sistema, de manera que todos los programas puedan acceder a ellos—esto es especialmente crucial en un sistema multiusurario. Para hacerlo, la administradora puede especificar varios paquetes libc en el campo locale-libcs de operating-system:

(use-package-modules base)

7 Las versiones 2.23 y posteriores de GNU libc simplemente ignorarán los datos de localización incompatibles, lo cual ya es un avance.
Capítulo 8: Configuración del sistema

(operating-system
 ;; ... (locale-libcs (list glibc-2.21 (canonical-package glibc)))

Este ejemplo llevaría a un sistema que contiene definiciones de localización tanto para libc 2.21 como para la versión actual de libc en /run/current-system/locale.

8.8 Servicios

Una parte importante de la preparación de una declaración operating-system es listar los servicios del sistema y su configuración (véase Sección 8.1 [Uso de la configuración del sistema], página 161). Los servicios del sistema típicamente son daemon lanzados cuando el sistema arranca, u otras acciones necesarias en ese momento—por ejemplo, configurar el acceso de red.

Guix tiene una definición amplia de “servicio” (véase Sección 8.17.1 [Composición de servicios], página 412), pero muchos servicios se gestionan por GNU Shepherd (véase Sección 8.17.4 [Servicios de Shepherd], página 420). En un sistema en ejecución, la orden herd le permite enumerar los servicios disponibles, mostrar su estado, arrancarlos y pararlos, o realizar otras acciones específicas (véase Sección “Jump Start” en The GNU Shepherd Manual). Por ejemplo:

# herd status

La orden previa, ejecutada como root, enumera los servicios actualmente definidos. La orden herd doc muestra una sinopsis del servicio proporcionado y sus acciones asociadas:

# herd doc nsdc

Run libc’s name service cache daemon (nsdc).

# herd doc nsdc action invalidate

invalidate: Invalidate the given cache—e.g., ‘hosts’ for host name lookups.

Las ordenes internas start, stop y restart tienen el efecto de arrancar, parar y reiniciar el servicio, respectivamente. Por ejemplo, las siguientes órdenes paran el servicio nsdc y reinician el servidor gráfico Xorg:

# herd stop nsdc

Service nsdc has been stopped.

# herd restart xorg-server

Service xorg-server has been stopped.

Service xorg-server has been started.

Las siguientes secciones documentan los servicios disponibles, comenzando con los servicios básicos, que pueden ser usados en una declaración operating-system.

8.8.1 Servicios base

El módulo (gnu services base) proporciona definiciones para los servicios básicos que se esperan en el sistema. Los servicios exportados por este módulo se enumeran a continuación.

%base-services

Esta variable contiene una lista de servicios básicos (véase Sección 8.17.2 [Tipos de servicios y servicios], página 414, para más información sobre los objetos servicio)
que se pueden esperar en el sistema: un servicio de ingreso al sistema (mingetty) en cada tty, syslogd, el daemon de la caché del servicio de nombres (nscd), el gestor de dispositivos udev, y más. 

Este es el valor predeterminado del campo services de las declaraciones operating-system. De manera habitual, cuando se personaliza el sistema, es deseable agregar servicios a %base-services, de esta forma:

```
(append (list (service avahi-service-type)
             (service openssh-service-type))
        %base-services)
```

### special-files-service-type

El servicio que establece “ficheros especiales” como /bin/sh; una instancia suya es parte de %base-services.

El valor asociado con servicios special-file-service-type debe ser una lista de tuplas donde el primer elemento es el “fichero especial” y el segundo elemento es su destino. El valor predeterminado es:

```
'(("/bin/sh", (file-append bash "/bin/sh")))
```

Si quiere añadir, digamos, /usr/bin/env a su sistema, puede cambiar su valor por:

```
'(("/bin/sh", (file-append bash "/bin/sh"))
   ("/usr/bin/env", (file-append coreutils "/bin/env")))
```

Ya que es parte de %base-services, puede usar modify-services para personalizar el conjunto de ficheros especiales (véase Sección 8.17.3 [Referencia de servicios], página 416). Pero una forma simple de añadir un fichero especial es usar el procedimiento extra-special-file (véase a continuación).

#### extra-special-file fichero destino

Usa destino como el “fichero especial” fichero.

Por ejemplo, la adición de las siguientes líneas al campo services de su declaración de sistema operativo genera /usr/bin/env como un enlace simbólico:

```
(extra-special-file "/usr/bin/env"
     (file-append coreutils "/bin/env"))
```

### host-name-service nombre

Devuelve un servicio que establece el nombre de máquina a nombre.

### console-font-service-type

Instal the given fonts on the specified ttys (fonts are per virtual console on the kernel Linux). The value of this service is a list of tty/font pairs. The font can be the name of a font provided by the kbd package or any valid argument to setfont, as in this example:

```
'(("tty1" . "LatGrkCyr-8x16")
  ("tty2" . ,(file-append
               font-tamzen
               "/share/kbd/consolefonts/TamzenForPowerline10x20.psf"))
  ("tty3" . ,(file-append
               font-terminus
               "/share/consolefonts/ter-132n"))) ; for HDPI
```
login-service config

[Procedimiento Scheme]

Devuelve un servicio para ejecutar el ingreso al sistema de acuerdo con config, un objeto <login-configuration>, que especifica el mensaje del día, entre otras cosas.

login-configuration

[Tipo de datos]

Este es el tipo de datos que representa la configuración del ingreso al sistema.

motd

Un objeto tipo-fichero que contiene el “mensaje del día”.

allow-empty-passwords? (predeterminado: #t)

Permite contraseñas vacías por defecto para que las primeras usuarias puedan ingresar en el sistema cuando la cuenta de “root” está recién creada.

mingetty-service config

[Procedimiento Scheme]

Devuelve un servicio para ejecutar mingetty de acuerdo con config, un objeto <mingetty-configuration>, que especifica el tty a ejecutar entre otras cosas.

mingetty-configuration

[Tipo de datos]

Este es el tipo de datos que representa la configuración de Mingetty, el cual proporciona la implementación predeterminada de ingreso al sistema en las consolas virtuales.

tty

El sistema de la consola en la que se ejecuta este Mingetty—por ejemplo, "tty1".

auto-login (predeterminado: #f)

Cuando sea verdadero, este campo debe ser una cadena que denote el nombre de usuario bajo el cual el sistema ingresa automáticamente. Cuando es #f, se deben proporcionar un nombre de usuario y una contraseña para ingresar en el sistema.

login-program (predeterminado: #f)

Debe ser #f, en cuyo caso se usa el programa predeterminado de ingreso al sistema (login de las herramientas Shadow), o una expresión-G que determine el nombre del programa de ingreso al sistema.

login-pause? (predeterminado: #f)

Cuando es #t en conjunció con auto-login, la usuaria deberá presionar una tecla para lanzar el shell de ingreso al sistema.

mingetty (predeterminado: mingetty)

El paquete Mingetty usado.

agetty-service config

[Procedure Scheme]

Devuelve un servicio para ejecutar agetty de acuerdo con config, un objeto <agetty-configuration>, que especifica el tty a ejecutar entre otras cosas.

agetty-configuration

[Tipo de datos]

Este es el tipo de datos que representa la configuración de agetty, que implementa el ingreso al sistema en las consolas virtuales y serie. Véase la página de manual agetty(8) para más información.

tty

El nombre de la consola en la que se ejecuta este agetty, como una cadena—por ejemplo, "ttyS0". Este parámetro es opcional, su valor predeterminado es un puerto serie razonable usado por el núcleo Linux.
Para ello, si hay un valor para una opción `agetty.tty` en la línea de órdenes del núcleo, agetty extraerá el nombre del dispositivo del puerto serie de allí y usará dicho valor.

Si no y hay un valor para la opción `console` con un tty en la línea de órdenes de Linux, agetty extraerá el nombre del dispositivo del puerto serie de allí y usará dicho valor.

En ambos casos, agetty dejará el resto de configuración de dispositivos serie (tasa de transmisión, etc.) sin modificar—a la espera de que Linux los haya fijado a los valores correctos.

**baud-rate** (predeterminado: `#f`)
Una cadena que contenga una lista separada por comas de una o más tasas de transmisión, en orden descendente.

**term** (predeterminado: `#f`)
Una cadena que contiene el valor usado para la variable de entorno `TERM`.

**eight-bits?** (predeterminado: `#f`)
En caso de ser `#t`, se asume que el tty permite el paso de 8 bits, y la detección de paridad está desactivada.

**auto-login** (predeterminado: `#f`)
Cuando se proporciona un nombre de ingreso al sistema, como una cadena, la usuaria especificada ingresará automáticamente sin solicitar su nombre de ingreso ni su contraseña.

**no-reset?** (predeterminado: `#f`)
En caso de ser `#t`, no reinicia los modos de control del terminal (cflags).

**host** (predeterminado: `#f`)
Acepta una cadena que contenga el "nombre_de_máquina_de_ingreso", que será escrito en el fichero `/var/run/utmpx`.

**remote?** (predeterminado: `#f`)
Cuando se fija a `#t` en conjunción con `host`, se añadirá una opción `-r "fakehost"` a la línea de órdenes del programa de ingreso al sistema especificado en `login-program`.

**flow-control?** (predeterminado: `#f`)
Cuando es `#t`, activa el control de flujo hardware (RTS/CTS).

**no-issue?** (predeterminado: `#f`)
Cuando es `#t`, el contenido del fichero `/etc/issue` no se mostrará antes de presentar el mensaje de ingreso al sistema.

**init-string** (predeterminada: `#f`)
Esto acepta una cadena que se enviará al tty o módem antes de mandar nada más. Puede usarse para inicializar un modem.

**no-clear?** (predeterminado: `#f`)
Cuando es `#t`, agetty no limpiará la pantalla antes de mostrar el mensaje de ingreso al sistema.
login-program (predeterminado: (file-append shadow "/bin/login"))
   Esto debe ser o bien una expresión-g que denote el nombre del programa
de ingreso al sistema, o no debe proporcionarse, en cuyo caso el valor
predeterminado es login del conjunto de herramientas Shadow.

local-line (predeterminado: #f)
   Controla el selector la línea CLOCAL. Acepta uno de estos tres símbolos
como parámetros, 'auto, 'always (siempre) o 'never (nunca). Si es #f,
el valor predeterminado elegido por agetty es 'auto.

extract-baud? (predeterminado: #f)
   Cuando es #t, instruye a agetty para extraer la tasa de transmisión de
los mensajes de estado producidos por ciertos tipos de módem.

skip-login? (predeterminado: #f)
   Cuando es #t, no solicita el nombre de la usuaria para el ingreso al sis-
tema. Puede usarse con el campo login-program para usar sistemas de
ingreso no estándar.

no-newline? (predeterminado: #f)
   Cuando es #t, no imprime una nueva línea antes de imprimir el fichero
/etc/issue.

login-options (predeterminadas: #f)
   Esta opción acepta una cadena que contenga opciones para proporcio-
nar al programa de ingreso al sistema. Cuando se use con login-program,
sea consciente de que una usuaria con malas intenciones podría inten-
tar introducir un nombre que contuviese opciones embebidas que serían
procesadas por el programa de ingreso.

login-pause (predeterminada: #f)
   Cuando es #t, espera la pulsación de cualquier tecla antes de mostrar el
mensaje de ingreso al sistema. Esto puede usarse en conjunción con auto-
login para ahorrar memoria lanzando cada shell cuando sea necesario.

chroot (predeterminado: #f)
   Cambia la raíz al directorio especificado. Esta opción acepta una ruta de
directorio como una cadena.

hangup? (predeterminado: #f)
   Usa la llamada del sistema Linux vhangup para colgar de forma virtual
el terminal especificado.

keep-baud? (predeterminado: #f)
   Cuando es #t, prueba a mantener la tasa de transmisión existente. Las ta-
sas de transmisión de baud-rate se usan cuando agetty recibe un carácter
BREAK.

timeout (predeterminado: #f)
   Cuando sea un valor entero, termina si no se pudo leer ningún nombre de
usuaria en timeout segundos.
detect-case? (predeterminado: #f)
Cuando es #t, activa la detección de terminales únicamente con mayúsculas. Esta configuración detectará un nombre de ingreso que contenga únicamente letras mayúsculas como un indicativo de un terminal con letras únicamente mayúsculas y activará las conversiones de mayúscula a minúscula. Tenga en cuenta que esto no permitirá caracteres Unicode.

wait-cr? (predeterminado: #f)
Cuando es #t, espera hasta que la usuaria o el modem envíen un carácter de retorno de carro o de salto de línea antes de mostrar /etc/issue o el mensaje de ingreso. Se usa de forma típica junto a la opción init-string.

no-hints? (predeterminado: #f)
Cuando es #t, no imprime avisos sobre el bloqueo numérico, las mayúsculas o el bloqueo del desplazamiento.

no-hostname? (predeterminado: #f)
El nombre de la máquina se imprime de forma predeterminada. Cuando esta opción es #t, no se mostrará ningún nombre de máquina.

long-hostname? (predeterminado: #f)
El nombre de máquina se imprime de forma predeterminada únicamente hasta el primer punto. Cuando esta opción es #t, se muestra el nombre completamente cualificado de la máquina mostrado por gethostname o getaddrinfo.

erase-characters (predeterminado: #f)
Esta opción acepta una cadena de caracteres adicionales que deben interpretarse como borrado del carácter anterior cuando la usuaria introduce su nombre de ingreso.

kill-characters (predeterminado: #f)
Esta opción acepta una cadena de que debe ser interpretada como “ignora todos los caracteres anteriores” (también llamado carácter “kill”) cuando la usuaria introduce su nombre de ingreso.

chdir (predeterminado: #f)
Esta opción acepta, como una cadena, una ruta de directorio que a la que se cambiará antes del ingreso al sistema.

delay (predeterminado: #f)
Esta opción acepta, como un entero, el número de segundos a esperar antes de abrir el tty y mostrar el mensaje de ingreso al sistema.

nice (predeterminado: #f)
Esta opción acepta, como un entero, el valor “nice” con el que se ejecutará el programa login.

extra-options (predeterminadas: '() )
Esta opción proporciona una "trampilla de escape" para que la usuaria proporcione parámetros de línea de órdenes adicionales a agetty como una lista de cadenas.
Capítulo 8: Configuración del sistema

kmscon-service-type config

[Procedimiento Scheme]
Devuelve un servicio que ejecuta kmscon (https://www.freedesktop.org/wiki/Software/kmscon) de acuerdo a config, un objeto \texttt{<kmscon-configuration>}, que especifica el tty sobre el que se ejecutará, entre otras cosas.

kmscon-configuration

[Tipo de datos]
Este es el tipo de datos que representa la configuración de Kmscon, que implementa el ingreso al sistema en consolas virtuales.

\begin{description}
\item[virtual-terminal]
El sistema de la consola en la que se ejecuta este Kmscon—por ejemplo, "tty1".
\item[login-program (predeterminado: \texttt{#\{string-append \$shadow \"/bin/login\}\})]
Una expresión-g que denota el programa de ingreso al sistema. El programa de ingreso al sistema predeterminado es login del conjunto de herramientas Shadow.
\item[login-arguments (predeterminados: \texttt{\'(-p\')}\)]
Una lista de parámetros para proporcionar a login.
\item[auto-login (predeterminado: \texttt{#f} \)]
Cuando se proporciona un nombre de ingreso al sistema, como una cadena, la usuaria especificada ingresará automáticamente sin solicitar su nombre de ingreso ni su contraseña.
\item[hardware-acceleration? (predeterminado: \texttt{#f} \)]
Determina si se usará aceleración hardware.
\item[kmscon (predeterminado: kmscon)\]
El paquete Kmscon usado.
\end{description}

nsccd-service [configuración] [#:glibc glibc]  
[#:name-services '()]

[Procedimiento Scheme]
Devuelve un servicio que ejecuta el daemon de la caché del servicio de nombres (nscd) con la configuración proporcionada—un objeto \texttt{<nscd-configuration>}. Véase Sección 8.11 [Selector de servicios de nombres], página 391, para un ejemplo.

Por conveniencia, el servicio nscd de Shepherd proporciona las siguientes acciones:

\begin{description}
\item[invalidate]
Esta invalida la caché dada. Por ejemplo, ejecutar:
\begin{verbatim}
herd invalidate nscd hosts
\end{verbatim}
invalida la caché de búsqueda de nombres de máquinas de nscd.
\item[statistics]
Ejecutar \texttt{herd statistics nscd} muestra información del uso nscd y la caché.
\end{description}

%nsccd-default-configuration

[Variable Scheme]
El valor del \texttt{<nscd-configuration>} predeterminado (véase a continuación) usado por \texttt{nscd-service}. Usa los espacios de caché definidos por \texttt{%nsccd-default-caches}; véase a continuación.
nsccd-configuration

Este tipo de datos representa la configuración del daemon de caché del servicio de nombres (nsccd).

name-services (predeterminados: '() )
Lista de paquetes que indican los servicios de nombres que serán visibles al nsccd—por ejemplo, (list nss-mdns).

glibc (predeterminada: glibc)
Paquete que denota la biblioteca C de GNU que proporciona la orden nsccd.

log-file (predeterminado: "/var/log/nsccd.log")
Nombre del fichero de registro de nsccd. Aquí es donde se almacena la salida de depuración cuando debug-level es estrictamente positivo.

dereg-level (predeterminado: 0)
Entero que indica el nivel de depuración. Números mayores significan que se registra más salida de depuración.

caches (predeterminadas: %nsccd-default-caches)
Lista de objetos <nsccd-cache> que indican cosas a mantener en caché; véase a continuación.

nsccd-cache

Tipo de datos que representa una base de datos de caché de nsccd y sus parámetros.

base de datos
Es un símbolo que representa el nombre de la base de datos de la que se actúa como caché. Se aceptan los valores passwd, group, hosts y services, que designan las bases de datos NSS correspondientes (véase Sección “NSS Basics” en The GNU C Library Reference Manual).

positive-time-to-live
negative-time-to-live (predeterminado: 20)
Un número que representa el número de segundos durante los que una búsqueda positiva o negativa permanece en la caché.

check-files? (predeterminado: #t)
Si se comprobará en busca de actualizaciones los ficheros que correspon-
dan con database.
Por ejemplo, cuando database es hosts, la activación de esta opción ins-
truye a nsccd para comprobar actualizaciones en /etc/hosts y tenerlas en cuenta.

persistent? (predeterminada: #t)
Determina si la caché debe almacenarse de manera persistente en disco.

shared? (predeterminado: #t)
Determina si la caché debe compartirse entre las usuarias.

max-database-size (predeterminado: 32 MiB)
Tamaño máximo en bytes de la caché de la base de datos.
%nscd-default-caches [Variable Scheme]

Lista de objetos <nscd-cache> usados por omisión por nscd-configuration (véase en la sección previa).

Activa el almacenamiento en caché persistente y agresivo de búsquedas de servicios y nombres de máquina. La última proporciona un mejor rendimiento en la búsqueda de nombres de máquina, resiliencia en caso de nombres de servidor no confiables y también mejor privacidad—a menudo el resultado de las búsquedas de nombres de máquina está en la caché local, por lo que incluso ni es necesario consultar servidores de nombres externos.

syslog-configuration [Tipo de datos]

Este tipo de datos representa la configuración del daemon syslog.

syslogd (predeterminado: #~(string-append #$inetutils "/libexec/syslogd"))

El daemon syslog usado.

config-file (predeterminado: %default-syslog.conf)

El fichero de configuración de syslog usado.

syslog-service config [Procedimiento Scheme]

Devuelve un servicio que ejecuta el daemon de syslog de acuerdo a config.

Véase Sección “syslogd invocation” en GNU Inetutils, para más información sobre la sintaxis del fichero de configuración.

guix-service-type [Variable Scheme]

El tipo de servicio que ejecuta el daemon de construcción, guix-daemon (véase Sección 2.5 [Invocación de guix-daemon], página 15). Su valor debe ser un registro guix-configuration como se describe a continuación.

guix-configuration [Tipo de datos]

Este tipo de datos representa la configuración del daemon de construcción de Guix. Véase Sección 2.5 [Invocación de guix-daemon], página 15, para más información.

guix (predeterminado: guix)

El paquete Guix usado.

build-group (predeterminado: "guixbuild")

El nombre del grupo de las cuentas de usuarias de construcción.

build-accounts (predeterminadas: 10)

Número de cuentas de usuarias de construcción a crear.

authorize-key? (predeterminado: #t)

Determina si se autoriza las claves de sustituciones listadas en authorized-keys—predeterminada la de ci.guix.gnu.org (véase Sección 4.3 [Sustituciones], página 44).

authorized-keys (predeterminadas: %default-authorized-guix-keys)

La lista de ficheros de claves autorizadas para importaciones de archivos, como una lista de expresiones-G que evalúan a cadenas (véase Sección 4.11 [Invocación de guix archive], página 63). Por defecto, contiene las de ci.guix.gnu.org (véase Sección 4.3 [Sustituciones], página 44).
use-substitutes? (predeterminado: #t)
Determina si se usarán sustituciones.

substitute-urls (predeterminado: %default-substitute-urls)
La lista de URLs donde se buscarán sustituciones por defecto.

max-silent-time (predeterminado: 0)
timeout (predeterminado: 0)
El número de segundos de silencio y el número de segundos de actividad respectivamente, tras los cuales el proceso de construcción supera el plazo. Un valor de cero proporciona plazos ilimitados.

log-compression (predeterminado: 'bzip2)
El tipo de compresión usado en los log de construcción—o bien gzip, o bien bzip2 o none.

extra-options (predeterminadas: '() )
Lista de opciones de línea de órdenes adicionales para guix-daemon.

log-file (predeterminado: "/var/log/guix-daemon.log")
Fichero al que se escriben la salida estándar y la salida estándar de error de guix-daemon.

http-proxy (predeterminado: #f)
El proxy HTTP que se usa para la descarga de derivaciones de salida fija y sustituciones.

tmpdir (predeterminado: #f)
Una ruta de directorio donde guix-daemon realiza las construcciones.

udev-service [#:udev eudev #:rules '()]
Ejecuta udev, que gestiona el contenido del directorio /dev de forma dinámica. Se pueden proporcionar reglas de udev como una lista de ficheros a través de la variable rules. Los procedimientos udev-rule y file->udev-rule de (gnu services base) simplifican la creación de dichos ficheros de reglas.

udev-rule [nombre-fichero contenido]
Devuelve un fichero de reglas de udev con nombre nombre-fichero que contiene las reglas definidas en el literal contenido.

En el ejemplo siguiente se define una regla para un dispositivo USB que será almacenada en el fichero 90-usb-cosa.rules. Esta regla ejecuta un script cuando se detecta un dispositivo USB con un identificador de producto dado.

(define %regla-ejemplo-udev
  (udev-rule
    "90-usb-cosa.rules"
    (string-append "ACTION=="add", "SUBSYSTEM=="usb", "ATTR{product}=="Ejemplo","RUN+="/ruta/al/ejecutable"')))

La orden herd rules udev, como root, devuelve el nombre del directorio que contiene todas las reglas udev activas.
Aquí se muestra cómo se puede extender el servicio udev-service predeterminado se puede extender con ella.

```
(operating-system
 ;; ... 
(services
 (modify-services %desktop-services
 (udev-service-type config =>
 (udev-configuration (inherit config)
 (rules (append (udev-configuration-rules config)
 (list %regla-ejemplo-udev)))))))))
```

**Procedimiento Scheme**

`file->udev-rule [nombre-fichero fichero]`  
Devuelve un fichero de udev con nombre `nombre-fichero` que contiene las reglas definidas en `fichero`, un objeto tipo-fichero.

El ejemplo siguiente muestra cómo podemos usar un fichero de reglas existente.

```
(use-modules (guix download) ;para url-fetch
 (guix packages) ;para origin 
 ...)

(define %reglas-android-udev
 (file->udev-rule
 "51-android-udev.rules"
 (let ((version "20170910"))
 (origin
 (method url-fetch)
 (uri (string-append "https://raw.githubusercontent.com/M0Rf30/
 android-udev-rules/" version "/51-android.rules"))
 (sha256
 (base32 "0lmmagpyb6xsq6zcr2w1cyx9qmjmqmajkvrdbhjx32gqf1d9is003")))
))
```

Adicionalmente, las definiciones de paquete Gui pueden ser incluidas en `rules` para extender las reglas udev con las definiciones encontradas bajo su subdirectorio `lib/udev/rules.d`. En vez del ejemplo previo de `file->udev-rule`, podríamos haber usado el paquete `android-udev-rules` que existe en Guix en el módulo (`gnu packages android`).

El siguiente ejemplo muestra cómo usar el paquete `android-udev-rules` para que la herramienta de Android `adb` pueda detectar dispositivos sin privilegios de root. También detalla cómo crear el grupo `adbusers`, el cual se requiere para el funcionamiento correcto de las reglas definidas dentro del paquete `android-udev-rules`. Para crear tal grupo, debemos definirlo tanto como parte de `supplementary-groups` de la declaración de nuestra cuenta de usuaria `user-account`, así como en el campo `groups` del registro `operating-system`.

```
(use-modules (gnu packages android) ;para android-udev-rules
 (gnu system shadow) ;para user-group
 ...)

(operating-system
 ;; ...
(users (cons (user-account
    ;; ...
    (supplementary-groups
      '("adbusers" ;for adb
                    "wheel" "netdev" "audio" "video"))))

(groups (cons (user-group (system? #t) (name "adbusers"))
      %base-groups))

services
  (modify-services %desktop-services
    (udev-service-type
      config =>
      (udev-configuration (inherit config)
        (rules (cons android-udev-rules
                (udev-configuration-rules config)))))

urandom-seed-service-type 
  [Variable Scheme]
  Almacena alguna entropía en %random-seed-file para alimentar /dev/urandom cuan-
  do se reinicia. También intenta alimentar /dev/urandom con /dev/hwrng durante el
  arranque, si /dev/hwrng existe y se tienen permisos de lectura.

%random-seed-file 
  [Variable Scheme]
  Es el nombre del fichero donde algunos bytes aleatorios son almacenados por el servi-
  cio urandom-seed-service para alimentar /dev/urandom durante el reinicio. Su valor
  predeterminado es /var/lib/random-seed.

gpm-service-type 
  [Variable Scheme]
  Este es el tipo de servicio que ejecuta GPM, el daemon de ratón de propósito general,
  que permite el uso del ratón en la consola Linux. GPM permite a las usuarias el uso
  del ratón en la consola, notablemente la selección, copia y pegado de texto.
  El valor para servicios de este tipo debe ser un objeto gpm-configuration (véase a
  continuación). Este servicio no es parte de %base-services.

gpm-configuration 
  [Tipo de datos]
  Tipo de datos que representa la configuración de GPM.

  opciones (predeterminadas: %default-gpm-options)
    Opciones de línea de órdenes proporcionadas a gpm. El conjunto prede-
    terminado de opciones instruye a gpm para esperar eventos de ratón en
    /dev/input/mice. Véase Sección “Command Line” en gpm manual, para
    más información.

  gpm (predeterminado: gpm)
    El paquete GPM usado.
guix-publish-service-type

Este es el tipo de servicio para guix publish (véase Sección 7.10 [Invocación de guix publish], página 149). Su valor debe ser un objeto guix-publish-configuration, como se describe a continuación.

Se asume que /etc/guix ya contiene el par de claves de firma como guix archive --generate-key lo crea (véase Sección 4.11 [Invocación de guix archive], página 63). Si no es el caso, el servicio fallará al arrancar.

guix-publish-configuration

Tipo de datos que representa la configuración del servicio guix publish.

guix (predeterminado: guix)
El paquete Guix usado.

port (predeterminado: 80)
El puerto TCP en el que se esperan conexiones.

host (predeterminado: "localhost")
La dirección de red (y, por tanto, la interfaz de red) en la que se esperarán conexiones. Use "0.0.0.0" para aceptar conexiones por todas las interfaces de red.

compression (predeterminada: '(("gzip" 3)))
Es una lista de tuplas método de compresión/nivel usadas para la compresión de sustituciones. Por ejemplo, para comprimir todas las sustituciones tanto con lzip a nivel 8 como con gzip a nivel 9, escriba:

'(("lzip" 7) ("gzip" 9))
El nivel 9 obtiene la mejor relación de compresión con un incremento en el uso del procesador, mientras que el nivel 1 realiza la compresión rápido. Una lista vacía desactiva completamente la compresión.

nar-path (predeterminado: "nar")
La ruta URL desde la que se pueden obtener “nars”. Véase Sección 7.10 [Invocación de guix publish], página 149, para más detalles.

cache (predeterminado: #f)
Cuando es #f, desactiva la caché y genera los archivos bajo demanda. De otro modo, debería ser el nombre de un directorio—por ejemplo, "/var/cache/guix/publish”—donde guix publish almacena los archivos y metadatos en caché listos para ser enviados. Véase Sección 7.10 [Invocación de guix publish], página 149, para más información sobre sus ventajas e inconvenientes.

workers (predeterminado: #f)
Cuando es un entero, es el número de hilos de trabajo usados para la caché; cuando es #f, se usa el número de procesadores. Véase Sección 7.10 [Invocación de guix publish], página 149, para más información.

ttl (predeterminado: #f)
Cuando es un entero, denota el tiempo de vida en segundos de los archivos publicados. Véase Sección 7.10 [Invocación de guix publish], página 149, para más información.
Capítulo 8: Configuración del sistema

rngd-service [#:rng-tools rng-tools] [#:device /dev/hwrng] [Procedimiento Scheme]

Devuelve un servicio que ejecuta el programa rngd de rng-tools para añadir device a la fuente de entropía del núcleo. El servicio emitirá un fallo si device no existe.

pam-limits-service [#:limits ()] [Procedimiento Scheme]

Devuelve un servicio que instala un fichero de configuración para el módulo pam_limits (http://linux-pam.org/Linux-PAM-html/sag-pam_limits.html). El procedimiento toma de manera opcional una lista de valores pam-limits-entry, que se pueden usar para especificar límites ulimit y límites de prioridad “nice” para sesiones de usuaria.

Las siguientes definiciones de límites establecen dos límites “hard” y “soft” para todas las sesiones de ingreso al sistema de usuarias pertenecientes al grupo realtime:

```
(pam-limits-service
 (list
  (pam-limits-entry "@realtime" 'both 'rtprio 99)
  (pam-limits-entry "@realtime" 'both 'memlock 'unlimited))))
```

La primera entrada incrementa la prioridad máxima de tiempo real para procesos sin privilegios; la segunda entrada elimina cualquier restricción sobre el espacio de direcciones que puede bloquearse en memoria. Estas configuraciones se usan habitualmente para sistemas de sonido en tiempo real.

8.8.2 Ejecución de tareas programadas

El módulo (gnu services mcron) proporciona una interfaz a GNU mcron, un daemon para ejecutar trabajos planificados de antemano (véase GNU mcron). GNU mcron es similar al daemon tradicional de Unix cron; la principal diferencia es que está implementado en Scheme Guile, que proporciona mucha flexibilidad cuando se especifica la planificación de trabajos y sus acciones.

El siguiente ejemplo define un sistema operativo que ejecuta las órdenes updatedb (véase Sección “Invoking updatedb” en Finding Files) y guix gc (véase Sección 4.5 [Invocación de guix gc], página 48) de manera diaria, así como la orden mkid como una usuaria sin privilegios (véase Sección “mkid invocation” en ID Database Utilities). Usa expresiones-g para introducir definiciones de trabajos que serán proporcionados a mcron (véase Sección 6.7 [Expresiones-G], página 108).

```
(use-modules (guix) (gnu) (gnu services mcron))
(use-package-modules base idutils)

(define trabajo-updatedb
 ;; Ejecuta 'updatedb' a las 3AM cada día. Aquí escribimos
 ;; las acciones del trabajo como un procedimiento Scheme.
 #"(job '(next-hour '(3))
  (lambda ()
    (execl (string-append #$findutils "/bin/updatedb")
      "updatedb"
      "--prunepaths=/tmp /var/tmp /gnu/store"))))
```
Capítulo 8: Configuración del sistema

(define trabajo-recolector-basura
;; Recolecta basura 5 minutos después de media noche,
;; todos los días. La acción del trabajo es una orden
;; del shell.
"#~(job "5 0 * * *" ; sintaxis de Vixie cron
"guix gc -F 1G")")

(define trabajo-idutils
;; Actualiza el índice de la base de datos como "carlos" a las
;; 12:15 y a las 19:15. Esto se ejecuta desde su directorio.
"#~(job '(next-minute-from (next-hour '(12 19)) '(15))
(string-append #$idutils "/bin/mkid src")
#:user "carlos")")

(operating-system
;; ...
(services (cons (service mcron-service-type
  (mcron-configuration
  (jobs (list trabajo-recolector-basura
          trabajo-updatedb
          trabajo-idutils)))
  %base-services)))

Para trabajos más complejos definidos en Scheme donde necesita control en el ámbito global, por ejemplo para introducir una forma use-modules, puede mover su código a un programa separado usando el procedimiento program-file del módulo (guix gexp) (véase Sección 6.7 [Expresiones-G], página 108). El siguiente ejemplo ilustra este caso.

(define %tarea-alerta-bateria
;; Pita cuando la carga de la batería es inferior a %CARGA-MIN
"#~(job
  '(next-minute (range 0 60 1))
  $(program-file
  "alerta-batería.scm"
  (with-imported-modules (source-module-closure
  "((guix build utils))")
  $(begin
    (define %carga-min 20)
    (use-modules (guix build utils)
      (ice-9 open-popen)
      (ice-9 ice-9)
      (ice-9 regex)
      (ice-9 textual-ports)
      (srfi srfi-2))
    (setenv "LC_ALL" "C") ; Procesado de cadenas en inglés
    (and-let* ((entrada (open-pipe* OPEN_READ
      file-append acpi "/bin/acpi")
    (salida (get-string-all entrada)
(m (string-match "Discharging, ([0-9]+)%" output))
(carga (string->number (match:substring m 1)))
((< carga %carga-min)))
(setenv "LC_ALL" "") ; Mensaje de salida traducido
(format #t "aviso: La carga de la batería es baja (~a%)" carga)
(invoke #$(file-append beep "/bin/beep") "-r5")))

Véase Sección “Guile Syntax” en GNU mcron, para más información sobre las especificaciones de trabajos de mcron. A continuación se encuentra la referencia del servicio mcron.

En un sistema en ejecución puede usar la acción schedule del servicio para visualizar los siguientes trabajos mcron que se ejecutarán:

# herd schedule mcron

El ejemplo previo enumera las siguientes cinco tareas que se ejecutarán, pero también puede especificar el número de tareas a mostrar:

# herd schedule mcron 10

mcron-service-type [Variable Scheme]
Este es el tipo del servicio mcron, cuyo valor es un objeto mcron-configuration.
Este tipo de servicio puede ser objeto de una extensión de servicio que le proporciona especificaciones de trabajo adicionales (véase Sección 8.17.1 [Composición de servicios], página 412). En otras palabras, es posible definir servicios que proporcionen trabajos mcron adicionales para su ejecución.

mcron-configuration [Tipo de datos]
Tipo de datos que representa la configuración de mcron.

mcron (predeterminado: mcron)
El paquete mcron usado.

jobs Es una lista de expresiones-G (véase Sección 6.7 [Expresiones-G], página 108), donde cada expresión-G corresponde a una especificación de trabajo de mcron (véase Sección “Syntax” en GNU mcron).

8.8.3 Rotación del registro de mensajes
Los ficheros de registro como los encontrados en /var/log tienden a crecer indefinidamente, de modo que es buena idea llevar a cabo una rotación de vez en cuando—es decir, archivar su contenido en ficheros distintos, posiblemente comprimidos. El módulo (gnu services admin) proporciona una interfaz con GNU Rot[t]log, una herramienta de rotación de registros (véase GNU Rot[t]log Manual).

El siguiente ejemplo define un sistema operativo que proporciona rotación de registros con la configuración predeterminada, para los ficheros de registro encontrados habitualmente.

(use-modules (guix) (gnu))
(use-service-modules admin mcron)
(use-package-modules base idutils)
(operating-system
  ;; ...
  (services (cons (service rottlog-service-type)
                  %base-services)))

rottlog-service-type [Variable Scheme]
Este es el tipo del servicio Rottlog, cuyo valor es un objeto rottlog-configuration.
Otros servicios pueden extenderlo con nuevos objetos log-rotation (véase a continuación), aumentando de dicho modo el conjunto de ficheros a rotar.
Este servicio puede definir trabajos de mcron (véase Sección 8.8.2 [Ejecución de tareas programadas], página 196) para ejecutar el servicio rottlog.

rottlog-configuration [Tipo de datos]
Tipo de datos que representa la configuración de rottlog.
rottlog (predeterminado: rottlog)
El paquete Rottlog usado.
rc-file (predeterminado: (file-append rottlog "/etc/rc")
rotations (predeterminadas: %default-rotations)
Una lista de objetos log-rotation como se define a continuación.
jobs Esta es una lista de expresiones-G donde cada expresión-G corresponde a una especificación de trabajo de mcron (véase Sección 8.8.2 [Ejecución de tareas programadas], página 196).

log-rotation [Tipo de datos]
Tipo de datos que representa la rotación de un grupo de ficheros de log.
Tomando el ejemplo del manual de Rottlog (véase Sección “Period Related File Examples” en GNU Rot[t]log Manual), una rotación de registros se podría definir de esta manera:

(log-rotation
  (frequency 'daily)
  (files '("/var/log/apache/*"))
  (options '("storedir apache-archives"
              "rotate 6"
              "notifempty"
              "nocompress")))

La lista de campos es como sigue:
frequency (predeterminada: 'weekly)
La frecuencia de rotación de logs, un símbolo.
files La lista de ficheros o patrones extendidos de fichero a rotar.
options (predeterminadas: '())
La lista de opciones de rottlog para esta rotación (véase Sección “Configuration parameters” en GNU Rot[t]log Manual).
Capítulo 8: Configuración del sistema

post-rotate (predeterminado: #f)

O bien #f, o bien una expresión-G que se ejecutará una vez la rotación se haya completado.

%default-rotations [Variable Scheme]

Especifica la rotación semanal de %rotated-files y de /var/log/guix-daemon.log.

%rotated-files [Variable Scheme]

La lista de ficheros controlados por syslog que deben ser rotados. De manera predeterminada es '("/var/log/messages" "/var/log/secure" "/var/log/maillog")

8.8.4 Servicios de red

El módulo (gnu services networking) proporciona servicios para configurar la interfaz de red.

dhcp-client-service-type [Variable Scheme]

Este es el tipo de los servicios que ejecutan dhcp, un cliente del protocolo de configuración dinámica de máquinas DHCP, en todas las interfaces de red no locales. Su valor es el paquete del cliente DHCP, isc-dhcp de manera predeterminada.

dhcpd-service-type [Procedimiento Scheme]

Este tipo define un servicio que ejecuta el daemon DHCP. Para crear un servicio de este tipo debe proporcionar un objeto <dhcpd-configuration>. Por ejemplo:

```
(service dhcpd-service-type
    (dhcpd-configuration
        (config-file (local-file "mi-dhcpd.conf"))
        (interfaces '("enp0s25"))))
```

dhcpd-configuration [Tipo de datos]

package (predeterminado: isc-dhcp)

El paquete que proporciona el daemon DHCP. Se espera que este paquete proporcione el daemon en/sbin/dhcpd de manera relativa a su directorio de salida. El paquete predeterminado es el servidor DHCP de ISC (https://www.isc.org/products/DHCP).

cfg-file (predeterminado: #f)


version (predeterminada: "4")

La versión DHCP usada. El servidor DHCP de ISC permite los valores “4”, “6” y “4o6”. Corresponden con las opciones -4, -6 y -4o6 del programa dhcpd. Véase man dhcpd para más detalles.

run-directory (predeterminado: "/run/dhcpd")

El directorio de ejecución usado. Durante la activación del servicio se creará en caso de no existir.
pid-file (predeterminado: "/run/dhcpd/dhcpd.pid")
   El fichero de PID usado. Corresponde con la opción -pf de dhcpd. Véase
   man dhcpd para más detalles.

interfaces (predeterminadas: '() )
   Los nombres de las interfaces de red en las que dhcpd debería esperar
   retransmisiones. Si la lista no está vacía, entonces sus elementos (que
   deben ser cadenas) se añadirán a la invocación de dhcpd cuando se inicie el
   daemon. Puede no ser necesaria la especificación explícita aquí de ninguna
   interfaz; véase man dhcpd para más detalles.

static-networking-service-type
   [Variable Scheme]
   Este es el tipo para las interfaces de red configuradas de forma estática.

static-networking-service interface ip [#:netmask #:gateway #:name-servers ’0 ] [#:requirement
   ’(udev) ] Devuelve un servicio que activa la interface con la dirección ip. Si netmask es
   verdadero, se usa como máscara de red. Si gateway es verdadero, se usa como cadena
   que especifica la pasarela predeterminada de la red. requirement puede usarse para
   especificar una dependencia en otro servicio antes de configurar la interfaz.

   Se puede llamar a este procedimiento varias veces, una por cada interfaz de red sobre
   la que se tenga interés. Internamente, lo que hace es extender static-networking-
   service-type con interfaces de red adicionales que gestionar.

   Por ejemplo:
   (static-networking-service "eno1" "192.168.1.82"
      #:gateway "192.168.1.2"
      #:name-servers ’("192.168.1.2")
   )

wicd-service [#:wicd wicd]  [Procedimiento Scheme]
   Devuelve un servicio que ejecuta Wicd (https://launchpad.net/wicd), un daemon
   de gestión de red que intenta simplificar la gestión de redes cableadas e inalámbricas.

   Este servicio añade wicd al perfil global, proporcionando varias ordenes destinadas a
   la interacción con el daemon y a la configuración de la red: wicd-client, una interfaz
   gráfica, y las interfaces de usuaria wicd-cli y wicd-curses.

modem-manager-service-type
   [Variable Scheme]
   Este es el tipo de servicio para el servicio ModemManager (https://wiki.gnome.
   org/Projects/ModemManager). El valor para este tipo de servicio es un registro
   modem-manager-configuration.

   Este servicio es parte de %desktop-services (véase Sección 8.8.7 [Servicios de escri-
   torio], página 237).

modem-manager-configuration
   [Tipo de datos]
   Tipo de datos que representa la configuración de ModemManager.

   modem-manager (predeterminado: modem-manager)
      El paquete de ModemManager usado.
Capítulo 8: Configuración del sistema

usb-modeswitch-service-type

This is the service type for the USB_ModeSwitch (https://www.draisberghof.de/usb_modeswitch/) service. The value for this service type is a usb-modeswitch-configuration record.

Cuando se conectan, algunos modem USB (y otros dispositivos USB) se presentan inicialmente como medios de almacenamiento de sólo-lectura y no como un modem. Deben cambiar de modo antes de poder usarse. El tipo de servicio USB_ModeSwitch instala reglas de udev para cambiar automáticamente de modo cuando se conecten estos dispositivos.

Este servicio es parte de %desktop-services (véase Sección 8.8.7 [Servicios de escritorio], página 237).

usb-modeswitch-configuration

Tipo de datos que representa la configuración de USB_ModeSwitch.

usb-modeswitch (predeterminado: usb-modeswitch)

El paquete USB_ModeSwitch que proporciona los binarios para el cambio de modo.

usb-modeswitch-data (predeterminado: usb-modeswitch-data)

El paquete que proporciona los datos de dispositivos y las reglas de udev usadas por USB_ModeSwitch.

config-file (predeterminado: #"(string-append #$usb-modeswitch:dispatcher "/etc/usb_modeswitch.conf")"

Fichero de configuración usado para el gestor de eventos (dispatcher) de USB_ModeSwitch. De manera predeterminada se usa el fichero que viene con USB_ModeSwitch, que deshabilita el registro en /var/log junto a otras configuraciones. Si se proporciona #f no se usa ningún fichero de configuración.

network-manager-service-type


Este servicio es parte de %desktop-services (véase Sección 8.8.7 [Servicios de escritorio], página 237).

network-manager-configuration

Tipo de datos que representa la configuración de NetworkManager.

network-manager (predeterminado: network-manager)

El paquete de NetworkManager usado.

dns (predeterminado: "default")

Modo de procesamiento para DNS, que afecta la manera en la que NetworkManager usa el fichero de configuración resolv.conf.

‘default’ NetworkManager actualizará resolv.conf para reflejar los servidores de nombres proporcionados por las conexiones activas actualmente.
'dnsmasq' NetworkManager ejecutará `dnsmasq` como una caché local del servicio de nombres, mediante un reenvío condicional si se encuentra conectada a una VPN, y actualiza posteriormente `resolv.conf` para apuntar al servidor de nombres local.

Con esta configuración puede compartir su conexión de red. Por ejemplo, cuando desee compartir su conexión de red a otro equipo a través de un cable Ethernet, puede abrir `nm-connection-editor` y configurar el método de la conexión cableada para IPv4 y IPv6 “Compartida con otros equipos” y restablecer la conexión (o reiniciar).

También puede configurar una conexión anfitrión-invitado a las máquinas virtuales de QEMU (véase Sección 3.8 [Instalación de Guix en una máquina virtual], página 33). Con una conexión anfitrión-invitado puede, por ejemplo, acceder a un servidor web que se execute en la máquina virtual (véase Sección 8.8.16 [Servicios Web], página 303) desde un navegador web en su sistema anfitrión, o conectarse a la máquina virtual a través de SSH (véase Sección 8.8.4 [Servicios de red], página 200). Para configurar una conexión anfitrión-invitado, ejecute esta orden una única vez:

```
nmcli connection add type tun \
connection.interface-name tap0 \
tun.mode tap tun.owner $(id -u) \
ipv4.method shared \
ipv4.addresses 172.28.112.1/24
```

Cada vez que arranque su máquina virtual de QEMU (véase Sección 8.16 [Ejecutar Guix en una máquina virtual], página 410), proporcione `-nic tap,ifname=tap0,script=no,downscript=no` a `qemu-system-...`

'none' NetworkManager no modificará `resolv.conf`.

`vpn-plugins` (predeterminados: '()')

Esta es la lista de módulos disponibles para redes privadas virtuales (VPN). Un ejemplo es el paquete `network-manager-openvpn`, que permite a NetworkManager la gestión de redes VPN a través de OpenVPN.

`connman-service-type` [Variable Scheme]

Este es el tipo de servicio para la ejecución de Connman (https://01.org.connman), un gestor de conexiones de red.

Su valor debe ser un registro `connman-configuration` como en este ejemplo:

```
(service connman-service-type 
  (connman-configuration 
   (disable-vpn? #t)))
```

Véase a continuación más detalles sobre `connman-configuration`.
Capítulo 8: Configuración del sistema

**connman-configuration** [Tipo de datos]
Tipo de datos que representa la configuración de connman.

- **connman** (predeterminado: connman)
  El paquete connman usado.

- **disable-vpn?** (predeterminado: #f)
  Cuando es verdadero, desactiva el módulo vpn de connman.

**wpa-supplicant-service-type** [Variable Scheme]
Este es el tipo de servicio para la ejecución de WPA supplicant (https://w1.fi/wpa_supplicant/), un daemon de identificación necesario para la identificación en redes WiFi o ethernet cifradas.

**wpa-supplicant-configuration** [Tipo de datos]
Tipo de datos que representa la configuración de WPA Supplicant.
Toma los siguientes parámetros:

- **wpa-supplicant** (predeterminado: wpa-supplicant)
  El paquete de WPA Supplicant usado.

- **dbus?** (predeterminado: #t)
  Si se escuchan o no peticiones en D-Bus.

- **pid-file** (predeterminado: "/var/run/wpa_supplicant.pid")
  Dónde se almacena el fichero con el PID.

- **interface** (predeterminado: #f)
  En caso de proporcionarse un valor, debe especificar el nombre de la interfaz de red que WPA supplicant controlará.

- **config-file** (predeterminado: #f)
  Fichero de configuración opcional usado.

- **extra-options** (predeterminadas: ()
  Lista de parámetros adicionales a pasar al daemon en la línea de órdenes.

**iptables-service-type** [Variable Scheme]
Este es el tipo de servicio para la aplicación de configuración de iptables. iptables es un entorno de trabajo para el filtrado de paquetes implementado por el núcleo Linux. Este servicio permite la configuración de iptables tanto para IPv4 como IPv6. Un ejemplo simple de cómo rechazar todas las conexiones entrantes excepto aquellas al puerto 22 de ssh se muestra a continuación.

```
(service iptables-service-type
 (iptables-configuration
  (ipv4-rules (plain-file "reglas.iptables" "*filter

 :INPUT ACCEPT
 :FORWARD ACCEPT
 :OUTPUT ACCEPT
 -A INPUT -p tcp --dport 22 -j ACCEPT
 -A INPUT -j REJECT --reject-with icmp-port-unreachable
 COMMIT
```
Capítulo 8: Configuración del sistema

```
(ipv6-rules (plain-file "reglas.ip6tables" "*filter
 :INPUT ACCEPT
 :FORWARD ACCEPT
 :OUTPUT ACCEPT
 -A INPUT -p tcp --dport 22 -j ACCEPT
 -A INPUT -j REJECT --reject-with icmp6-port-unreachable
 COMMIT
 ")))
```

### iptables-configuration

El tipo de datos que representa la configuración de iptables.

- **iptables** (predeterminado: `iptables`)
  El paquete iptables que proporciona `iptables-restore` y `ip6tables-restore`.

- **ipv4-rules** (predeterminado: `%iptables-accept-all-rules`)
  Las reglas de iptables usadas. Se le proporcionarán a `iptables-restore`. Puede ser cualquier objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108).

- **ipv6-rules** (predeterminadas: `%iptables-accept-all-rules`)
  Las reglas de ip6tables usadas. Se le proporcionarán a `ip6tables-restore`. Puede ser cualquier objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108).

### nftables-service-type

Es el tipo de servicio para levantar una configuración de nftables. nftables es un proyecto de netfilter que quiere reemplazar los entornos ya existentes iptables, ip6tables, arptables y ebtables. Proporciona un entorno de filtrado de paquetes nuevo, una utilidad `nft` de espacio de usuaria nueva y una capa de compatibilidad con iptables. El servicio viene con un conjunto de reglas predeterminado `%default-nftables-ruleset` que rechaza todas las conexiones entrantes excepto las del puerto 22. Para usarlo, simplemente escriba:

```
(service nftables-service-type)
```

### nftables-configuration

El tipo de datos que representa la configuración de nftables.

- **package** (predeterminado: `nftables`)
  El paquete nftables que proporciona nft.

- **ruleset** (predeterminados: `%default-nftables-ruleset`)
  El conjunto de reglas de nftables usado. Puede ser cualquier objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108).

### ntp-service-type

This is the type of the service running the Network Time Protocol (NTP) ([https://www.ntp.org](https://www.ntp.org)) daemon, `ntpd`. The daemon will keep the system clock synchronized with that of the specified NTP servers.
El valor de este servicio es un objeto \texttt{ntpd-configuration}, como se describe a continuación.

\textbf{ntpd-configuration} \hfill [Tipo de datos]

Este es el tipo de datos para la configuración del servicio NTP.

\textbf{servers} (predeterminados: \%\texttt{ntp-servers})

La lista de servidores (registros \texttt{<ntp-server>}) con los que la herramienta ntpd se sincronizará. Véase la información sobre el tipo de datos \texttt{ntp-server} a continuación.

\textbf{allow-large-adjustment?} (predeterminado: \#t)

Esto determina si se le permite a ntpd realizar un ajuste inicial de más de 1000 segundos.

\textbf{ntp} (predeterminado: \texttt{ntp})

El paquete NTP usado.

\textbf{\%\texttt{ntp-servers}} \hfill [Variable Scheme]

Lista de nombres de máquinas usadas como servidores NTP predeterminados. Son servidores del NTP Pool Project (https://www.ntppool.org/en/).

\textbf{ntp-server} \hfill [Tipo de datos]

Tipo de datos que representa la configuración de un servidor NTP.

\textbf{type} (predeterminado: \texttt{'server})

El tipo del servidor NTP, proporcionado como un símbolo. Puede ser \texttt{'pool, 'server, 'peer, 'broadcast} o \texttt{'manycastclient}.

\textbf{address} La dirección del servidor, como una cadena.

\textbf{options} Opciones de NTPD usadas en ese servidor específico, proporcionada como una lista de nombres de opciones y/o tuplas de nombre y valor. El siguiente ejemplo define un servidor con el que se usan las opciones \texttt{iburst} y \texttt{prefer}, así como \texttt{version 3} y un tiempo de 16 segundos para \texttt{maxpoll}.

\begin{verbatim}
(ntp-server
 (type 'server)
 (address "miservidor.ntp.server.org")
 (options '(iburst (version 3) (maxpoll 16) prefer)))))
\end{verbatim}

\textbf{openntpd-service-type} \hfill [Procedimiento Scheme]

Ejecuta ntpd, el daemon del protocolo de tiempo en red (NTP), implementado por OpenNTPD (http://www.openntpd.org). El daemon mantendrá el reloj del sistema sincronizado con el de los servidores proporcionados.

\begin{verbatim}
(service
 openntpd-service-type
 (openntpd-configuration
 (listen-on '"127.0.0.1" "::1")
 (sensor "udcf0 correction 70000")
 (constraint-from "www.gnu.org")
 (constraints-from "https://www.google.com/")
 (allow-large-adjustment? #t))))
\end{verbatim}
Capítulo 8: Configuración del sistema

%openntpd-servers
Esta variable es una lista de las direcciones de servidores definidos en %ntp-servers.

openntpd-configuration
El ejecutable openntpd usado.

listen-on
Una lista de direcciones IP o nombres de máquina en los que el daemon ntpd debe escuchar conexiones.

query-from
Una lista de direcciones IP locales que el daemon ntpd debe usar para consultas salientes.

sensor
Especifica una lista de dispositivos de sensores de tiempo de ntpd que no. ntpd escuchará cada sensor que realmente exista e ignora los que no. Véase la documentación de las desarrolladoras originales (https://man.openbsd.org/ntpd.conf) para más información.

server
Especifica una lista de direcciones IP o nombres de máquina de servidores NTP con los que sincronizarse.

servers
Una lista de direcciones IP o nombres de máquina con los que el daemon ntpd se debe sincronizar.

collection-from
ntpd puede configurarse para que solicite la fecha a través del campo “Date” de servidores HTTPS en los que se confíe a través de TLS. Esta información de tiempo no se usa por precisión pero actúa como una condición verificada, por tanto reduciendo el impacto de ataques mediante la intervención del tráfico con servidores NTP no verificados. Especifica una lista de URL, direcciones IP o nombres de máquina de servidores HTTPS que proporcionarán la condición.

constraints-from
Como en constraint-from, proporciona una lista de URL, direcciones IP o nombres de máquina de servidores HTTP para proporcionar la condición. En caso de que el nombre de máquina resuelva en múltiples direcciones IP, ntpd calculará la condición mediana de todas ellas.

allow-large-adjustment?
Determina si se permite a ntpd realizar un ajuste inicial de más de 180 segundos.

inetd-service-type
Este servicio ejecuta el daemon inetd (véase Sección “inetd invocation” en GNU Inetutils). inetd escucha conexiones en sockets de internet, e inicia bajo demanda el programa servidor cuando se realiza una conexión en uno de esos sockets.
El valor de este servicio es un objeto `inetd-configuration`. El ejemplo siguiente configura el daemon `inetd` para proporcionar el servicio `echo` implementado por él mismo, así como un servicio smtp que reenvíe el tráfico smtp por ssh a un servidor `servidor-smtp` tras la pasarela máquina:

```scheme
(service
  inetd-service-type
  (inetd-configuration
    (entries (list
      (inetd-entry
        (name "echo")
        (socket-type 'stream)
        (protocol "tcp")
        (wait? #f)
        (user "root")
      )
      (inetd-entry
        (node "127.0.0.1")
        (name "smtp")
        (socket-type 'stream)
        (protocol "tcp")
        (wait? #f)
        (user "root")
        (program (file-append openssh "/bin/ssh")
          (arguments
            "ssh" "-qT" "-i" "/ruta/de/la/clave_ssh"
            "-W" "servidor-smtp:25" "usuaria@maquina"))))
  )))
)
```

A continuación se proporcionan más detalles acerca de `inetd-configuration`.

`inetd-configuration` [Tipo de datos]
Tipo de datos que representa la configuración de `inetd`.

`program` (predeterminado: (file-append inetutils "/libexec/inetd")
El ejecutable `inetd` usado.

`entries` (predeterminadas: '())
Una lista de entradas de servicio de `inetd`. Cada entrada debe crearse con el constructor `inetd-entry`.

`inetd-entry` [Tipo de datos]
Tipo de datos que representa una entrada en la configuración de `inetd`. Cada entrada corresponde a un socket en el que `inetd` escuchará a la espera de peticiones.

`node` (predeterminado: #f)
Cadena opcional, una lista separada por comas de direcciones locales que `inetd` debería usar cuando se escuche para este servicio. Véase Sección “Configuration file” en GNU Inetutils para una descripción completa de todas las opciones.

`name` Una cadena, el nombre debe corresponder con una entrada en `/etc/services`.
socket-type
Puede ser 'stream, 'dgram, 'raw, 'rdm o 'seqpacket.

protocol Una cadena, debe corresponder con una entrada en /etc/protocols.

wait? (predeterminado: #t)
Si inetd debe esperar la salida del servidor antes de reiniciar la escucha de nuevas peticiones de servicio.

user Una cadena que contiene el nombre (y, opcionalmente, el grupo) de la usuaria como la que se deberá ejecutar el servidor. El nombre de grupo se puede especificar en un sufixo, separado por dos puntos o un punto normal, es decir "usuaria", "usuaria:grupo" o "usuaria.grupo".

program (predeterminado: "internal")
El programa servidor que recibirá las peticiones, o "internal" si inetd debería usar un servicio implementado internamente.

arguments (predeterminados: '())
Una lista de cadenas u objetos “tipo-fichero”, que serán los parámetros del programa servidor, empezando con el parámetro 0, es decir, el nombre del programa en sí mismo. Para los servicios internos de inetd, esta entrada debe ser '() o '("internal")).

Véase Sección “Configuration file” en GNU Inetutils, para una información más detallada sobre cada campo de la configuración.

tor-service-type [Variable Scheme]
Este es el tipo para un servicio que ejecuta el daemon de red anónima Tor (https://torproject.org). El servicio se configura mediante un registro <tor-configuration>. De manera predeterminada, el daemon Tor se ejecuta como la usuaria sin privilegios tor, que es miembro del grupo tor.

tor-configuration [Tipo de datos]
tor (predeterminado: tor)
El paquete que proporciona el daemon Tor. Se espera que este paquete proporcione el daemon en bin/tor de manera relativa al directorio de su salida. El paquete predeterminado es la implementación del Proyecto Tor (https://www.torproject.org).

config-file (predeterminado: (plain-file "empty" ""))
El fichero de configuración usado. Se agregará al final del fichero de configuración predeterminado, y se proporcionará el fichero de configuración resultante a tor a través de su opción -f. Puede ser cualquier objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108). Véase man tor para detalles sobre la sintaxis del fichero de configuración.

hidden-services (predeterminados: '())
La lista de registros de servicios ocultos <hidden-service> usados. Para cada servicio oculto que añada en esta lista, se activará la configuración apropiada para su activación en el fichero de configuración predeterminado. Puede crear registros <hidden-service> de manera conveniente
mediante el uso del procedimiento `tor-hidden-service` descrito a continuación.

**socks-socket-type** (predeterminado: `tcp`)
El tipo socket predeterminado que Tor debe usar para su socket SOCKS. Debe ser `tcp` o `unix`. Si es `tcp`, Tor escuchará en el puerto TCP 9050 de la interfaz local (es decir, localhost) de manera predeterminada. Si es `unix`, tor escuchará en el socket de dominio de UNIX `/var/run/tor/socks-sock`, que tendrá permisos de escritura para miembros del grupo `tor`.

Si desea personalizar el socket SOCKS de manera más detallada, mantenga `socks-socket-type` con su valor predeterminado de `tcp` y use `config-file` para modificar el valor predeterminado proporcionando su propia opción `SocksPort`.

**tor-hidden-service nombre relación** [Procedimiento Scheme]
Define un servicio oculto Tor llamado `nombre` y que implementa la `relación`. `relación` es una lista de tuplas puerto/máquina, como:

```text
((22 "127.0.0.1:22")
 (80 "127.0.0.1:8080"))
```

En este ejemplo, el puerto 22 del servicio oculto se asocia con el puerto 22 local, y el puerto 80 se asocia con el puerto 8080 local.

Esto crea un directorio `/var/lib/tor/hidden-services/nombre`, donde el fichero `hostname` contiene el nombre de máquina .onion para el servicio oculto.


El módulo (gnu services rsync) proporciona los siguientes servicios:

Puede ser que desee un daemon rsync si tiene ficheros que desee tener disponibles de modo que cualquiera (o simplemente usted) pueda descargar ficheros existentes o subir nuevos ficheros.

**rsync-service-type** [Variable Scheme]

```
(service rsync-service-type)
```

Véase a continuación para detalles sobre `rsync-configuration`.

**rsync-configuration** [Tipo de datos]
Tipo de datos que representa la configuración para `rsync-service`.

*package* (predeterminado: rsync)
Paquete rsync usado.

*port-number* (predeterminado: 873)
Puerto TCP en el que rsync escucha conexiones entrantes. Si el puerto es menor a 1024, rsync necesita iniciarse como root, tanto usuaria como grupo.
Capítulo 8: Configuración del sistema

pid-file (predeterminado: "/var/run/rsyncd/rsyncd.pid")
Nombre del fichero donde rsync escribe su PID.

lock-file (predeterminado: "/var/run/rsyncd/rsyncd.lock")
Nombre del fichero donde rsync escribe su fichero de bloqueo.

log-file (predeterminado: "/var/log/rsyncd.log")
Nombre del fichero donde rsync escribe su fichero de registros.

use-chroot? (predeterminado: #t)
Si se debe usar “chroot” en el directorio compartido de rsync.

share-path (predeterminado: /srv/rsync)
Localización del directorio compartido de rsync.

share-comment (predeterminado: "Rsync share")
Comentario del directorio compartido de rsync.

read-only? (predeterminado: #f)
Permisos de lectura/escritura en el directorio compartido.

timeout (predeterminado: 300)
Plazo de la E/S en segundos.

user (predeterminada: "root")
Propietaria del proceso rsync.

group (predeterminado: "root")
Grupo del proceso rsync.

uid (predeterminado: "rsyncd")
Nombre o ID de usuario bajo la cual se efectúan las transferencias desde y hacia el módulo cuando el daemon se ejecuta como root.

gid (predeterminado: "rsyncd")
Nombre o ID de grupo que se usa cuando se accede al módulo.

Es más, (gnu services ssh) proporciona los siguientes servicios.

lsh-service [#:host-key "/etc/lsh/host-key"]
[Procedimiento Scheme]
[#:daemonic? #t] [#:interfaces ()] [#:port-number 22]
[#:x11-forwarding? #t] [#:tcp/ip-forwarding? #t]
[#:password-authentication? #t] [#:public-key-authentication? #t]
[#:initialize? #t]
Ejecuta el programa lshd de lsh para escuchar en el puerto port-number. host-key debe designar a un fichero que contiene la clave de la máquina, y que sea legible únicamente por root.

Cuando initialize? es verdadero, crea automáticamente una semilla una clave de máquina una vez se active el servicio si no existen todavía. Puede tomar un tiempo prolongado y necesita interacción.

Cuando initialize? es falso, es cuestión de la usuaria la inicialización del generador aleatorio (véase Sección “lsh-make-seed” en LSH Manual), la creación de un par de claves y el almacenamiento de la clave privada en el fichero host-key (véase Sección “lshd basics” en LSH Manual).

Cuando interfaces está vacío, lshd escucha conexiones en todas las interfaces de red; en otro caso, interfaces debe ser una lista de nombres de máquina o direcciones.

allow-empty-passwords? especifica si se aceptará el ingreso al sistema con una contraseña vacía y root-login? especifica si se acepta el ingreso al sistema como root.

El resto de opciones deberían ser autodescriptivas.

**openssh-service-type**

Este es el tipo para el daemon de shell seguro OpenSSH (http://www.openssh.org), sshd. Su valor debe ser un registro openssh-configuration como en este ejemplo:

```
(service openssh-service-type
 (oppenssh-configuration
  (x11-forwarding? #t)
  (permit-root-login 'without-password)
  (authorized-keys
   '((("alicia" ,(local-file "alicia.pub"))
     ("rober" ,(local-file "rober.pub"))))))

Véase a continuación detalles sobre openssh-configuration.

Este servicio se puede extender con claves autorizadas adicionales, como en este ejemplo:

```
(service-extension openssh-service-type
 (const '(("carlos",
      ,(local-file "carlos.pub"))))
```

**openssh-configuration**

Este es el registro de configuración para sshd de OpenSSH.

pid-file (predeterminado: "/var/run/sshd.pid")

Nombre del fichero donde sshd escribe su PID.

port-number (predeterminado: 22)

Puerto TCP en el que sshd espera conexiones entrantes.

permit-root-login (predeterminado: #f)

Este fichero determina si y cuando se permite el ingreso al sistema como root. Si es #f, el ingreso como root no está permitido; si es #f está permitido. Si es el símbolo 'without-password, se permite el ingreso al sistema como root pero no con identificación basada en contraseña.

allow-empty-passwords? (predeterminado: #f)

Cuando es verdadero, las usuarias con contraseñas vacías pueden ingresar en el sistema. Cuando es falso, no pueden.
password-authentication? (predeterminado: #t)
Cuando es verdadero, las usuarias pueden acceder al sistema con su contraseña. En caso contrario, tienen otros métodos de identificación.

public-key-authentication? (predeterminado: #t)
Cuando es verdadero, las usuarias pueden acceder al sistema mediante el uso de clave pública para su identificación. Cuando es falso, las usuarias tienen que usar otros métodos de identificación.

Las claves públicas autorizadas se almacenan en `/ssh/authorized_keys`. Se usa únicamente por la versión 2 del protocolo.

x11-forwarding? (predeterminado: #f)
Cuando verdadero, la retransmisión de conexiones del cliente gráfico X11 está desactivada—en otras palabras, las opciones `-X` y `-Y` de `ssh` funcionarán.

allow-agent-forwarding? (predeterminado: #t)
Si se permite la retransmisión del agente de claves.

allow-tcp-forwarding? (predeterminado: #t)
Si se permite la retransmisión TCP.

gateway-ports? (predeterminado: #f)
Si se permiten los puertos pasarela.

challenge-response-authentication? (predeterminado: #f)
Especifica si la identificación mediante respuesta de desafío está permitida (por ejemplo, a través de PAM).

use-pam? (predeterminado: #t)
Permite el uso de la interfaz de módulos de identificación conectables (PAM). Si es #t se activará la identificación PAM mediante el uso de `challenge-response-authentication?` y `password-authentication?`, además del proceso de los módulos de cuenta usuario y de sesión de PAM en todos los tipos de identificación.

Debido a que la identificación mediante respuesta de desafío de PAM tiene un rol equivalente a la identificación por contraseña habitualmente, debería desactivar `challenge-response-authentication?` o `password-authentication?`.

print-last-log? (predeterminado: #t)
Especifica si `sshd` debe imprimir la fecha y hora del último acceso al sistema de la usuaria cuando una usuaria ingresa interactivamente.

subsystems (predeterminados: `'(("sftp" "internal-sftp"))`)
Configura subsistemas externos (por ejemplo, el daemon de transmisión de ficheros).

Esta es una lista de listas de dos elementos, cada una de las cuales contiene el nombre del subsistema y una orden (con parámetros opcionales) para ejecutar tras petición del subsistema.
La orden `internal-sftp` implementa un servidor SFTP dentro del mismo proceso. De manera alternativa, se puede especificar la orden `sftp-server`:

```
(service openssh-service-type
  (openssh-configuration
   (subsystems
    `(('"sftp" ,(file-append openssh "/libexec/sftp-server"))))))
```

`accepted-environment` (predeterminado: `()' )

Una lista de cadenas que describe qué variables de entorno pueden ser exportadas.

Cada cadena obtiene su propia línea. Véase la opción `AcceptEnv` en `man sshd_config`.

This example allows ssh-clients to export the `COLORTERM` variable. It is set by terminal emulators, which support colors. You can use it in your shell’s resource file to enable colors for the prompt and commands if this variable is set.

```
(service openssh-service-type
  (openssh-configuration
   (accepted-environment '("COLORTERM")))
```

`authorized-keys` (predeterminadas: `()' )

Esta es la lista de claves autorizadas. Cada elemento de la lista es un nombre de usuario seguido de uno o más objetos “tipo-fichero” que representan claves públicas SSH. Por ejemplo:

```
(openssh-configuration
  (authorized-keys
   `(('"rekado" ,(local-file "rekado.pub"))
     ('"chris" ,(local-file "chris.pub"))
     ('"root" ,(local-file "rekado.pub"),(local-file "chris.pub"))))
```

registra las claves públicas especificadas para las cuentas rekado, chris y root.

Se pueden especificar claves autorizadas adicionales a través de `service-extension`.

Tenga en cuenta que esto no interfere con el uso de `~/.ssh/authorized_keys`.

`log-level` (predeterminado: `'info`)

Es un símbolo que especifica el nivel de detalle en los registros: `quiet`, `fatal`, `error`, `info`, `verbose`, `debug`, etc. Véase la página del manual de `sshd_config` para la lista completa de los nombres de nivel.

`extra-content` (predeterminado: `""`)

Este campo puede usarse para agregar un texto arbitrario al fichero de configuración. Es especialmente útil para configuraciones elaboradas que no se puedan expresar de otro modo. Esta configuración, por ejemplo, generalmente desactivaría el ingreso al sistema como root, pero lo permite para una dirección IP específica:
Capítulo 8: Configuración del sistema

(ssh-configuration
(extra-content "\nMatch Address 192.168.0.1
PermitRootLogin yes")

dropbear-service [config] [Procedimiento Scheme]
Por ejemplo, para especificar un servicio Dropbear que escuche en el puerto 1234, añada esta llamada al campo services de su sistema operativo:

(dropbear-service (dropbear-configuration (port-number 1234)))

dropbear-configuration [Tipo de datos]
Este tipo de datos representa la configuración del daemon Dropbear SSH.

dropbear (predeterminado: dropbear)
El paquete de Dropbear usado.

port-number (predeterminado: 22)
Puerto TCP donde el daemon espera conexiones entrantes.

syslog-output? (predeterminado: #t)
Determina si se envía la salida a syslog.

pid-file (predeterminado: "/var/run/dropbear.pid")
El nombre de fichero del fichero de PID del daemon.

root-login? (predeterminado: #f)
Si se permite el ingreso al sistema como root.

allow-empty-passwords? (predeterminado: #f)
Si se permiten las contraseñas vacías.

password-authentication? (predeterminado: #t)
Determina si se usará identificación basada en contraseña.

%facebook-host-aliases [Variable Scheme]
Esta variable habitualmente se usa en el campo hosts-file de una declaración operating-system (véase Sección 8.2 [Referencia de operating-system], página 168):

(use-modules (gnu) (guix))

(operating-system
 (host-name "micompu")
 ;; ...

Capítulo 8: Configuración del sistema

(hosts-file

;; Crea un fichero /etc/hosts file con alias para "localhost"
;; y "micompu", así como los servidores de facebook.
(plain-file "hosts"

  (string-append (local-host-aliases host-name)

    %facebook-host-aliases))))

Este mecanismo puede impedir a los programas que se ejecutan localmente, como navegadores Web, el acceso a Facebook.

El módulo (gnu services avahi) proporciona la siguiente definición.

avahi-service-type [Variable Scheme]
Es el servicio que ejecuta avahi-daemon, un servidor mDNS/DNS-SD a nivel del sistema que permite el descubrimiento de servicios y la búsqueda de nombres de máquina “sin configuración/cero configuración” (véase https://avahi.org/). Su valor debe ser un registro zero-configuration—véase a continuación.

This service extends the name service cache daemon (nscd) so that it can resolve .local host names using nss-mdns (https://0pointer.de/lennart/projects/nss-mdns/). Véase Sección 8.11 [Selector de servicios de nombres], página 391, for information on host name resolution.

De manera adicional, añade el paquete avahi al perfil del sistema de manera que ordenes como avahi-browse estén disponibles de manera directa.

avahi-configuration [Tipo de datos]
Tipo de datos que representa la configuración de Avahi.

host-name (predeterminado: #f)
Si es diferente de #f, se usa como el nombre de máquina a publicar para esta máquina; en otro caso, usa el nombre actual de la máquina.

publish? (predeterminado: #t)
Cuando es verdadero, permite la publicación (retransmisión) de nombres de máquina y servicios a través de la red.

publish-workstation? (predeterminado: #t)
Cuando es verdadero, avahi-daemon publica el nombre de máquina y la dirección IP a través de mDNS en la red local. Para ver los nombres de máquina publicados en su red local, puede ejecutar:

  avahi-browse _workstation._tcp

wide-area? (predeterminado: #f)
Cuando es verdadero, se permite DNS-SD sobre DNS unicast.

ipv4? (predeterminado: #t)
ipv6? (predeterminado: #t)

domains-to-browse (predeterminado: ’())
Esta es la lista de dominios a explorar.
openvswitch-service-type [Variable Scheme]
Este es el tipo del servicio Open vSwitch (https://www.openvswitch.org), cuyo valor debe ser un objeto openvswitch-configuration.

openvswitch-configuration [Tipo de datos]
Tipo de datos que representa la configuración de Open vSwitch, un switch virtual multicapa que está diseñado para permitir una automatización masiva en la red a través de extensión programática.

package (predeterminado: openvswitch)
   El objeto paquete de Open vSwitch.

pagekite-service-type [Variable Scheme]
El tipo de servicio para el servicio PageKite (https://pagekite.net), una solución de encaminado para hacer servidores de la red local visibles públicamente, incluso detrás de cortafuegos restrictivos o NAT sin redirección de puertos. El valor para este servicio es un registro pagekite-configuration.

Este es un ejemplo que expone los daemon HTTP y SSH locales:

(service pagekite-service-type
  (pagekite-configuration
    (kites '("http:@kitename:localhost:80:@kitesecret"
             "raw/22:@kitename:localhost:22:@kitesecret")
    (extra-file "/etc/pagekite.rc")))

pagekite-configuration [Tipo de datos]
Tipo de datos que representa la configuración de PageKite.

package (predeterminado: pagekite)
   El objeto paquete de PageKite.

 kitename (predeterminado: #f)
   Nombre de PageKite para la identificación con el servidor de fachada.

 kitesecret (predeterminado: #f)
   Secreto compartido para la comunicación con el servidor. Probablemente debería almacenarlo dentro extra-file en vez de aquí.

 frontend (predeterminado: #f)
   Conecta al servidor de fachada de PageKite con este nombre en vez de al servicio de pagekite.net.

 kites (predeterminados: '("http:@kitename:localhost:80:@kitesecret")

 extra-file (predeterminado: #f)
   Fichero adicional de configuración que debe leerse, el cual se espera que sea creado de forma manual. Úselo para añadir opciones adicionales y gestionar secretos compartidos fuera de banda.
8.8.5 Sistema X Window

El sistema gráfico X Window—específicamente Xorg—se proporciona en el módulo (gnu services xorg). Újese que no existe un procedimiento xorg-service. En vez de eso, el servidor X se inicia por el gestor de ingreso al sistema, de manera predeterminada el gestor de acceso de GNOME (GDM).

GDM por supuesto que permite a las usuarias ingresar al sistema con gestores de ventanas y entornos de escritorio distintos a GNOME; para aquellas que usan GNOME, GDM es necesario para características como el bloqueo automático de pantalla.

Para usar X11, debe instalar al menos un gestor de ventanas—por ejemplo los paquetes windowmaker o openbox—, preferiblemente añadiendo el que desee al campo packages de su definición de sistema operativo (véase Sección 8.2 [Referencia de operating-system], página 168).

\textbf{gdm-service-type} \\
Este es el tipo para el gestor de acceso de GNOME (https://wiki.gnome.org/Projects/GDM) (GDM), un programa que gestiona servidores gráficos y maneja de forma gráfica el ingreso al sistema de usuarias. Su valor debe ser  ungdm-configuration (véase a continuación).

GDM busca tipos de sesión descritos por los ficheros .desktop en /run/current-system/profile/share/xsessions y permite a las usuarias seleccionar una sesión en la pantalla de ingreso. Paquetes como gnome, xfce y i3 proporcionan ficheros .desktop; su adición a la lista global de paquetes hace que estén automáticamente disponibles en la pantalla de ingreso al sistema.

Además, se respetan los ficheros ~/.xsession. Cuando esté disponible, ~/.xsession debe ser un ejecutable que inicie un gestor de ventanas y/o otros clientes de X.

\textbf{gdm-configuration} \\
[Tipo de datos]

\textbf{auto-login?} (predeterminado: #f) \\
\textbf{default-user} (predeterminado: #f) \\
Cuando auto-login? es falso, GDM presenta una pantalla de ingreso.
Cuando auto-login? es verdadero, GDM ingresa directamente al sistema como default-user.

\textbf{debug?} (predeterminado: #f) \\
Cuando tiene valor verdadero, GDM escribe los mensajes de depuración en su registro.

\textbf{gnome-shell-assets} (predeterminados: ...) \\
Lista de activos de GNOME Shell necesarios para GDM: tema de iconos, fuentes, etc.cc

\textbf{xorg-configuration} (predeterminada: (xorg-configuration)) \\
Configuración del servidor gráfico Xorg.

\textbf{xsession} (predeterminado: (xinitrc)) \\
Guión a ejecutar antes de iniciar una sesión X.

\textbf{dbus-daemon} (predeterminado: dbus-daemon-wraper) \\
El nombre de fichero del ejecutable dbus-daemon.
Capítulo 8: Configuración del sistema

**gdm** (predeterminado: **gdm**)  El paquete GDM usado.

**slim-service-type**  [Variable Scheme]
Este es el tipo para el gestor de ingreso al sistema gráfico para X11 SLiM.

Como GDM, SLiM busca tipos de sesión descritos por ficheros **.desktop** y permite a las usuarias la selección de sesión en la pantalla de ingreso al sistema mediante el uso de **F1**. También respeta los ficheros **~/.xsession**.

Al contrario que GDM, SLiM no lanza las sesiones de las usuarias en terminales virtuales diferentes al usado para el ingreso, lo que significa que únicamente puede iniciar una sesión gráfica. Si desea ejecutar varias sesiones gráficas de manera simultánea, debe añadir múltiples servicios de SLiM a los servicios de su sistema. El ejemplo siguiente muestra cómo sustituir el servicio GDM predeterminado con dos servicios de SLiM en tty7 y tty8.

```lisp
(use-modules (gnu services)
  (gnu services desktop)
  (gnu services xorg)
  (srfi srfi-1)) ;para 'remove'

(operating-system
 ; ; ...
(services (cons* (service slim-service-type (slim-configuration
  (display "::0")
  (vt "vt7")))

  (service slim-service-type (slim-configuration
  (display "::1")
  (vt "vt8")))

  (remove (lambda (service)
    (eq? (service-kind service) gdm-service-type))
    %desktop-services))))

**slim-configuration**  [Tipo de datos]
Tipo de datos que representa la configuración de **slim-service-type**.

**allow-empty-passwords?** (predeterminado: #t)
Si se permite el ingreso al sistema con contraseñas vacías.

**auto-login?** (predeterminado: #f)
**default-user** (predeterminado: "")
Cuando **auto-login?** es falso, SLiM presenta una pantalla de ingreso.
Cuando **auto-login?** es verdadero, SLiM ingresa en el sistema directamente como **default-user**.

**theme** (predeterminado: **%default-slim-theme**)
**theme-name** (predeterminado: **%default-slim-theme-name**)
El tema gráfico usado y su nombre.
auto-login-session (predeterminado: #f)
Si es verdadero, debe ser el nombre del ejecutable a arrancar como la sesión predeterminada—por ejemplo, (file-append windowmaker "/bin/windowmaker").
Si es falso, se usará una sesión de las descritas en uno de los ficheros .desktop disponibles en /run/current-system/profile y ~/.guix-profile.
Nota: Debe instalar al menos un gestor de ventanas en el perfil del sistema o en su perfil de usuario. En caso de no hacerlo, si auto-login-session es falso, no podrá ingresar al sistema.

xorg-configuration (predeterminada (xorg-configuration))
Configuración del servidor gráfico Xorg.

display (predeterminada: ":0")
La pantalla en la que se iniciará el servidor gráfico Xorg.
vt (predeterminado: "vt7")
El terminal virtual (VT) en el que se iniciará el servidor gráfico Xorg.
xauth (predeterminado: xauth)
El paquete XAuth usado.
shepherd (predeterminado: shepherd)
El paquete de Shepherd usado para la invocación de halt y reboot.
sessreg (predeterminado: sessreg)
El paquete sessreg usado para el registro de la sesión.
slim (predeterminado: slim)
El paquete SLiM usado.

%default-theme [Variable Scheme]
%default-theme-name [Variable Scheme]
El tema predeterminado de SLiM y su nombre.
sddm-configuration [Tipo de datos]
Este es el tipo de datos para la configuración del servicio sddm.
display-server (predeterminado: "x11")
Selecciona el servidor de pantalla usado para el saludo. Los valores validos son "11" o "wayland".
numlock (predeterminado: "on")
Se aceptan los valores "on", "off" y "none".
halt-command (predeterminado #~(string-append #$shepherd "/sbin/halt"))
Orden a ejecutar para parar el sistema.
reboot-command (predeterminado #~(string-append #$shepherd "/sbin/reboot"))
Orden a ejecutar para reiniciar el sistema.
theme (predeterminado "maldive")
    Theme to use. Default themes provided by SDDM are "elarun", "maldives" or "maya".

themes-directory (predeterminado
    "/run/current-system/profile/share/sddm/themes")
    Directorio en el que buscar temas.

faces-directory (predeterminado
    "/run/current-system/profile/share/sddm/faces")
    Directorio en el que buscar caras.

default-path (predeterminado "/run/current-system/profile/bin")
    El valor predeterminado del PATH.

minimum-uid (predeterminado: 1000)
    UID mínimo mostrado en SDDM y al que se le permite el acceso.

maximum-uid (predeterminado: 2000)
    UID máximo mostrado en SDDM.

remember-last-user? (predeterminado #t)
    Recuerda la última usuaria.

remember-last-session? (predeterminado #t)
    Recuerda la última sesión.

hide-users (predeterminado "")
    Nombres de usuaria a ocultar de la pantalla de inicio de SDDM.

hide-shells (predeterminado 
    "(string-append #$shadow "/sbin/nologin")")
    Las usuarias que tengan alguno de los shell enumerados se ocultarán de la pantalla de inicio de SDDM.

session-command (predeterminado 
    "(string-append #$sddm 
    "/share/sddm/scripts/wayland-session")")
    Guión a ejecutar antes de iniciar una sesión wayland.

sessions-directory (predeterminado
    "/run/current-system/profile/share/wayland-sessions")
    Directorio en el que buscar ficheros desktop que inicien sesiones wayland.

xorg-configuration (predeterminada (xorg-configuration))
    Configuración del servidor gráfico Xorg.

xauth-path (predeterminado 
    "(string-append #$xauth 
    "/bin/xauth")")
    Ruta de xauth.

xephyr-path (predeterminado 
    "(string-append #$xorg-server 
    "/bin/Xephyr")")
    Ruta de Xephyr.

xdisplay-start (predeterminado 
    "(string-append #$sddm 
    "/share/sddm/scripts/Xsetup")")
    Guión a ejecutar tras iniciar xorg-server.
Capítulo 8: Configuración del sistema

xdisplay-stop (predeterminado ~(string-append "$sddm
"/share/sddm/scripts/Xstop"))
Guión a ejecutar antes de parar xorg-server.

xsession-command (predeterminado: xinitrc)
Guión a ejecutar antes de iniciar una sesión X.

xsessions-directory (predeterminado:
"/run/current-system/profile/share/xsessions")
Directorio para buscar ficheros desktop que inicien sesiones X.

minimum-vt (predeterminado: 7)
VT mínimo usado.

auto-login-user (predeterminado "")
Usuraria usada para el ingreso automático en el sistema.

auto-login-session (predeterminado "")
Fichero desktop usado para el ingreso automático en el sistema.

relogin? (predeterminado #f)
Volver a ingresar en el sistema tras salir.

sddm-service-type
This is the type of the service to run the SDDM display manager (https://github.com/sddm/sddm). Its value must be a sddm-configuration record (see below).
Este es un ejemplo de su uso:
   (service sddm-service-type
     (sddm-configuration
       (auto-login-user "alicia")
       (auto-login-session "xfce.desktop")))

sddm-configuration
Este tipo de datos representa la configuración del gestor de ingreso al sistema SDDM.
Los campos disponibles son:
sddm (predeterminado: sddm)
El paquete SDDM usado.
display-server (predeterminado: "x11")
Debe ser o bien "x11" o bien "wayland".

auto-login-user (predeterminado: "")
Si no está vacío, es la cuenta de usuraria que ingresará al sistema de manera automática.

auto-login-session (predeterminada: "")
Si no está vacío, es el nombre de fichero .desktop usado en el ingreso automático al sistema.

xorg-configuration
Este tipo de datos representa la configuración del servidor gráfico Xorg. Fíjese que no existe un servicio Xorg; en vez de eso, el servidor X es iniciado por un “gestor de
Capítulo 8: Configuración del sistema 223

pantalla” como GDM, SDDM y SLiM. Por tanto, la configuración de estos gestores de pantalla agrega un registro \texttt{xorg-configuration}.

\textbf{modules} (predeterminados: \%default-xorg-modules)

Esta es la lista de \texttt{paquetes de módulos} cargados por el servidor Xorg—por ejemplo, \texttt{xf86-video-vesa, xf86-input-keyboard}, etcétera.

\textbf{fonts} (predeterminadas: \%default-xorg-fonts)

Es una lista de directorios de tipografías a añadir a la \texttt{ruta de tipografías} del servidor.

\textbf{drivers} (predeterminados: ‘() )

Debe ser o bien la lista vacía, en cuyo caso Xorg selecciona el controlador gráfico automáticamente, o una lista de nombres de controladores que se intentarán en el orden especificado—por ejemplo, ("modesetting" "vesa").

\textbf{resolutions} (predeterminadas: ‘() )

Cuando \texttt{resolutions} es la lista vacía, Xorg selecciona una resolución de pantalla adecuada. En otro caso, debe ser una lista de resoluciones—por ejemplo, ((1024 768) (640 480)).

\textbf{keyboard-layout} (predeterminada: \#f )

Si es \#f, Xorg usa la distribución de teclado predeterminada—normalmente inglés de EEUU (“qwerty”) para un teclado de PC de 105 teclas.

En otro caso, debe ser un objeto \textbf{keyboard-layout} que especifique la distribución de teclado usada para la ejecución de Xorg. Véase Sección 8.6 [Distribución de teclado], página 179, para más información sobre cómo especificar la distribución de teclado.

\textbf{extra-config} (predeterminada: ‘() )

Es una lista de cadenas u objetos añadida al final del fichero de configuración. Se usa para proporcionar texto adicional para ser introducido de forma literal en el fichero de configuración.

\textbf{server} (predeterminado: \texttt{xorg-server})

Este es el paquete que proporciona el servidor Xorg.

\textbf{server-arguments} (predeterminados: \%default-xorg-server-arguments)

Es la lista de parámetros de línea de órdenes que se proporcionarán al servidor X. El valor predeterminado es \texttt{-nolisten tcp}.

\textbf{set-xorg-configuration config} \hspace{1cm} [Procedimiento Scheme]

[\texttt{tipo-de-servicio-del-gestor-de-pantalla}]

Le dice al gestor de pantalla (de tipo \texttt{tipo-de-servicio-del-gestor-de-pantalla}) que use \texttt{config}, un registro \texttt{<xorg-configuration>}. Debido a que la configuración de Xorg se embebe en la configuración del gestor de ingreso en el sistema—por ejemplo, \texttt{gdm-configuration}—este procedimiento proporciona un atajo para establecer la configuración de Xorg.
xorg-start-command [config]  

[Procedimiento Scheme]

Devuelve un script startx en el que los módulos, las tipografías, etcétera, especificadas en config están disponibles. El resultado debe usarse en lugar de startx.

Habitualmente el servidor X es iniciado por un gestor de ingreso al sistema.

screen-locker-service paquete [programa]  

[Procedimiento Scheme]

Añade paquete, un paquete para un bloqueador de sesión o un salvapantallas cuya orden es programa, al conjunto de programas setuid y añade una entrada PAM para él. Por ejemplo:

```
(screen-locker-service xlockmore "xlock")
```

permite usar el viejo XlockMore.

### 8.8.6 Servicios de impresión

El módulo (gnu services cups) proporciona una definición de servicio Guix para el servicio de impresión CUPS. Para añadir el soporte de impresión en un sistema Guix, añada un servicio cups-service en su definición de sistema operativo:

**cups-service-type**  

[Variable Scheme]

El tipo de servicio para el servidor de impresión CUPS. Su valor debe ser una configuración de CUPS válida (véase a continuación). Para usar la configuración predeterminada, simplemente escriba:

```
(service cups-service-type)
```

La configuración de CUPS controla los aspectos básicos de su instalación de CUPS: sobre qué interfaces se escuchará, qué hacer si falla un trabajo de impresión, cuanta información registrar, etcétera. Para realmente añadir una impresora, debe visitar la URL http://localhost:631, o usar una herramienta como los servicios de configuración de impresión de GNOME. De manera predeterminada, la configuración de un servicio CUPS generará un certificado auto-firmado en caso de ser necesario, para ofrecer conexiones seguras con el servidor de impresión.

Suponiendo que desee activar la interfaz Web de CUPS, y también añadir el paquete escpr para comunicarse con impresoras Epson y el paquete hplip-minimal para hacerlo con impresoras HP. Puede hacerlo directamente, de esta manera (debe usar el módulo (gnu packages cups)):

```
(service cups-service-type
  (cups-configuration
    (web-interface? #t)
    (extensions
      (list cups-filters escpr hplip-minimal))))
```

Fíjese: Si desea usar la interfaz gráfica basada en Qt5 que viene con el paquete hplip se le sugiere que instale el paquete hplip, o bien en su configuración del sistema operativo o bien como su usuaria.

A continuación se encuentran los parámetros de configuración disponibles. El tipo de cada parámetro antecede la definición del mismo; por ejemplo, ‘string-list foo’ indica que el parámetro foo debe especificarse como una lista de cadenas. También existe la posibilidad
de especificar la configuración como una cadena, si tiene un fichero cupsd.conf antiguo que quiere trasladar a otro sistema; véase el final para más detalles.

Los campos disponibles de cups-configuration son:

package cups
El paquete CUPS.

lista-paquetes extensions
Controladores y otras extensiones al paquete CUPS.

ficheros-conf files-configuration
Configuración sobre dónde escribir los registros, qué directorios usar para las colas de impresión y parámetros de configuración privilegiados relacionados.

Los campos disponibles de files-configuration son:

ruta-registro access-log
Define el nombre de fichero del registro de acceso. La especificación de un nombre de fichero en blanco desactiva la generación de registros de acceso. El valor stderr hace que las entradas de registro se envíen al fichero de la salida estándar de error cuando el planificador se ejecute en primer plano, o al daemon de registro del sistema cuando se ejecute en segundo plano. El valor syslog envía las entradas de registro al daemon de registro del sistema. El nombre de servidor puede incluirse en los nombres de fichero mediante el uso de la cadena %s, como en /var/log/cups/%s-access_log.

El valor predeterminado es ‘”/var/log/cups/access_log”’.

nombre-fichero cache-dir
Donde CUPS debe almacenar los datos de la caché.

El valor predeterminado es ‘”/var/cache/cups”’.

string config-file-perm
Especifica los permisos para todos los ficheros de configuración que escriba el planificador.

Tenga en cuenta que los permisos para el fichero printers.conf están configurados actualmente de modo que únicamente la usuaria del planificador (habitualmente root) tenga acceso. Se hace de esta manera debido a que las URI de las impresoras a veces contienen información sensible sobre la identificación que no debería conocerse de manera general en el sistema. No hay forma de desactivar esta característica de seguridad.

El valor predeterminado es ‘”0640”’.

ruta-registro error-log
Define el nombre de fichero del registro de error. La especificación de un nombre de fichero en blanco desactiva la generación de registros de error. El valor stderr hace que las entradas del registro se envíen al fichero de la salida de error estándar cuando el planificador se ejecute en primer plano, o al daemon de registro del sistema cuando se ejecute en segundo plano. El valor syslog provoca que las entradas del registro se envíen al daemon de registro del sistema. El
nombre del servidor puede incluirse en los nombres de fichero mediante el uso de la cadena %s, como en /var/log/cups/%s-error_log.
El valor predeterminado es “/var/log/cups/error_log”.

**string fatal-errors** [parámetro de files-configuration]
Especifica qué errores son fatales, los cuales provocan la salida del planificador.
El tipo de cadenas son:

- **none**  
  Ningún error es fatal.

- **all**  
  Todos los errores a continuación son fatales.

- **browse**  
  Los errores de la inicialización de exploración son fatales, por ejemplo las conexiones fallidas al daemon DNS-SD.

- **config**  
  Los errores de sintaxis en el fichero de configuración son fatales.

- **listen**  
  Los errores de escucha o de puertos son fatales, excepto fallos IPv6 en la red local o en direcciones any.

- **log**  
  Los errores de creación o escritura en el fichero de registros son fatales.

- **permissions**  
  La mala configuración de los permisos de los ficheros al inicio son fatales, por ejemplo certificados TLS compartidos y ficheros de claves con permisos de escritura para todo el mundo.

El valor predeterminado es “all -browse”.

**boolean file-device?** [parámetro de files-configuration]
Especifica si el pseudo-dispositivo del fichero puede usarse para nuevas colas de impresión. Siempre se permite la URI file:///dev/null.
El valor predeterminado es ‘#f’

**string group** [parámetro de files-configuration]
Especifica el nombre de grupo o ID usado para la ejecución de programas externos.
El valor predeterminado es ‘"lp”’.

**string log-file-perm** [parámetro de files-configuration]
Especifica los permisos para todos los ficheros de registro que el planificador escriba.
El valor predeterminado es ‘"0644”’.

**ruta-registro page-log** [parámetro de files-configuration]
Define el nombre de fichero del registro de páginas. La especificación de un nombre de fichero en blanco desactiva la generación de registro de páginas. El valor stderr hace que las entradas del registro se envíen al fichero de la salida de error cuando el planificador se ejecute en primer plano, o al daemon de registro del sistema cuando se ejecuten en segundo plano. El valor syslog provoca que las entradas del registro se envíen al daemon de registro del sistema. El nombre
del servidor puede incluirse en los nombres de fichero mediante el uso de la cadena \%s, como en /var/log/cups/%s-page_log.

El valor predeterminado es ‘"/var/log/cups/page_log"’.

**string remote-root** [parámetro de files-configuration]
Es especifica el nombre de la usuaria asociado con accesos sin identificación por parte de clientes que digan ser la usuaria root. La usuaria predeterminada es remroot.

El valor predeterminado es ‘"remroot"’.

**nombre-fichero request-root** [parámetro de files-configuration]
Es especifica el directorio que contiene los trabajos de impresión y otros datos de peticiones HTTP.

El valor predeterminado es ‘"/var/spool/cups"’.

**aislamiento sandboxing** [parámetro de files-configuration]
Es especifica el nivel de seguridad del aislamiento (sandbox) que se aplica sobre los filtros de impresión, motores y otros procesos lanzados por el planificador; o bien relaxed o bien strict. Esta directiva únicamente tiene uso actualmente en macOS.

El valor predeterminado es ‘strict’.

**nombre-fichero server-keychain** [parámetro de files-configuration]
Es especifica la localización de los certificados TLS y las claves privadas. CUPS buscará claves públicas y privadas en este directorio: un fichero .crt para certificados codificados con PEM y los correspondientes fichero .key para las claves privadas codificadas con PEM.

El valor predeterminado es ‘"/etc/cups/ssl"’.

**nombre-fichero server-root** [parámetro de files-configuration]
Es especifica el directorio que contiene los ficheros de configuración del servidor.

El valor predeterminado es ‘"/etc/cups"’.

**boolean sync-on-close?** [parámetro de files-configuration]
Es especifica si el planificador llama fsync(2) tras la escritura de los ficheros de configuración o estado.

El valor predeterminado es ‘#f’

**cadenas-separadas-por-espacios** [parámetro de files-configuration]

**system-group**
Es especifica el o los grupos usados para la identificación del grupo @SYSTEM.

**nombre-fichero temp-dir** [parámetro de files-configuration]
Es especifica el directorio donde se escriben los ficheros temporales.

El valor predeterminado es ‘"/var/spool/cups/tmp"’.
string user  [parámetro de files-configuration]  
Especifica el nombre de usuario o ID usado para la ejecución de programas externos.
El valor predeterminado es ‘"lp"'.

string set-env  [parámetro de files-configuration]  
Establece el valor de la variable de entorno especificada que se proporcionará a los procesos lanzados.
El valor predeterminado es ‘"variable value"'.

nivel-registro-acceso  [parámetro de cups-configuration]  
   access-log-level  
Especifica el nivel de registro para el fichero AccessLog. El nivel config registra la adición, borrado o modificación de impresoras y clases, y el acceso y modificación de los ficheros de configuración. El nivel actions registra cuando los trabajos de impresión se envían, mantienen a la espera, liberan, modifican o cancelan, además de todas las condiciones de config. El nivel all registra todas las peticiones.
El valor predeterminado es ‘actions’.

boolean auto-purge-jobs?  [parámetro de cups-configuration]  
Especifica si se purgarán los datos del histórico de trabajos de manera automática cuando ya no son necesarios para las cuotas.
El valor predeterminado es ‘#f’

lista-cadenas-separada-comas  [parámetro de cups-configuration]  
   browse-dns-sd-sub-types  
Especifica una lista de subtipos de DNS-SD anunciados por cada impresora compartida. Por ejemplo, ‘"_cups" "_print"’ le dirá a los clientes de red que se implementa tanto la compartición con CUPS como IPP Everywhere.
El valor predeterminado es ‘"_cups"'.

protocolos browse-local-protocols  [parámetro de cups-configuration]  
Especifica qué protocolos deben usarse para compartir las impresoras locales.
El valor predeterminado es ‘dnssd’.

boolean browse-web-if?  [parámetro de cups-configuration]  
Especifica si se anuncia la interfaz web de CUPS.
El valor predeterminado es ‘#f’

boolean browsing?  [parámetro de cups-configuration]  
Especifica si se anuncian las impresoras compartidas.
El valor predeterminado es ‘#f’

string classification  [parámetro de cups-configuration]  
Especifica la clasificación de seguridad del servidor. Se puede usar cualquier nombre válido para anunciar la identificación del nivel de seguridad en inglés, incluyendo
"classified", "confidential", "secret", "topsecret" y "unclassified"8, o el anuncio puede omitirse para desactivar las funciones de impresión segura.

El valor predeterminado es ‘”’.

**boolean classify-override?**  [parámetro de cups-configuration]

Especifica si las usuarias pueden modificar la clasificación (página de portada) de trabajos de impresión individuales mediante el uso de la opción job-sheets.

El valor predeterminado es ‘#f’.

**tipo-id-pred default-auth-type**  [parámetro de cups-configuration]

Especifica el tipo de identificación usado por omisión.

El valor predeterminado es ‘Basic’.

**cifrado-pred default-encryption**  [parámetro de cups-configuration]

Especifica si se usará cifrado para peticiones con identificación.

El valor predeterminado es ‘Required’.

**string default-language**  [parámetro de cups-configuration]

Especifica el idioma predeterminado usado para el texto y contenido de la web.

El valor predeterminado es ‘”en”’.

**cadena default-paper-size**  [parámetro de cups-configuration]

Especifica el tamaño predeterminado del papel para colas de impresión nuevas. ‘"Auto”’ usa el valor predeterminado de la localización, mientras que ‘"None”’ especifica que no hay un tamaño de papel predeterminado. Los nombres de tamaños específicos habitualmente son ‘"Letter”’ o ‘"A4”’.9

El valor predeterminado es ‘"Auto”’.

**string default-policy**  [parámetro de cups-configuration]

E especifica la política de acceso usada por omisión.

El valor predeterminado es ‘"default”’.

**boolean default-shared?**  [parámetro de cups-configuration]

Especifica si las impresoras locales se comparten de manera predeterminada.

El valor predeterminado es ‘#t’.

**entero-no-negativo dirty-clean-interval**  [parámetro de cups-configuration]

Especifica el retraso para la actualización de los ficheros de configuración y estado, en segundo. Un valor de 0 hace que la actualización se lleve a cabo tan pronto sea posible, en algunos milisegundos habitualmente.

El valor predeterminado es ‘30’.

---

8 NdT: La traducción de los niveles sería “clasificado”, “confidencial”, “secreto”, “alto secreto” y “desclasi- sificado”, respectivamente

9 NdT: ‘Letter’ es el formato estándar de ANSI, de 215,9x279,4 milímetros de tamaño, mientras que A4 es el formato estándar de ISO, de 210x297 milímetros de tamaño.
Capítulo 8: Configuración del sistema

**política-error error-policy** [parámetro de cups-configuration]
Especifica qué hacer cuando ocurra un error. Los valores posibles son *abort-job*, que descartará el trabajo de impresión fallido; *retry-job*, que intentará llevar de nuevo a cabo el trabajo en un momento posterior; *retry-current-job*, que reintenta el trabajo que falló de manera inmediata; y *stop-printer*, que para la impresora.
El valor predeterminado es ‘*stop-printer*’.

**entero-no-negativo filter-limit** [parámetro de cups-configuration]
Especifica el coste máximo de filtros que se ejecutan de manera concurrente, lo que puede usarse para minimizar problemas de recursos de disco, memoria y procesador. Un límite de 0 desactiva la limitación del filtrado. Una impresión media con una impresora no-PostScript necesita una limitación del filtrado de 200 más o menos. Una impresora PostScript necesita cerca de la mitad (100). Establecer un límite por debajo de estos valores limitará de forma efectiva al planificador a la ejecución de un único trabajo de impresión en cualquier momento.
El valor predeterminado es ‘0’.

**entero-no-negativo filter-nice** [parámetro de cups-configuration]
Especifica la prioridad de planificación de los filtros que se ejecuten para la impresión de un trabajo. El valor de “nice” va desde 0, la mayor prioridad, a 19, la menor prioridad.
El valor predeterminado es ‘0’.

**búsqueda-nombres-máquina** [parámetro de cups-configuration]
**host-name-lookups**
Especifica si se realizarán las búsquedas inversas en las conexiones de clientes. La opción *double* instruye a cupsd para que verifique que el nombre de máquina al que resuelve la dirección corresponde con la dirección devuelta por dicho nombre de máquina. Las búsquedas dobles también evitan que clientes con direcciones sin registrar se conecten a su servidor. Configure esta opción con #t o *double* únicamente si es absolutamente necesario.
El valor predeterminado es ‘#f’.

**entero-no-negativo job-kill-delay** [parámetro de cups-configuration]
Especifica el número de segundos a esperar antes de terminar los filtros y el motor asociados con un trabajo cancelado o puesto en espera.
El valor predeterminado es ‘30’.

**entero-no-negativo job-retry-interval** [parámetro de cups-configuration]
Especifica el intervalo entre los reintentos de trabajos en segundos. Se usa de manera habitual en colas de fax pero también puede usarse con colas de impresión normales cuya política de error sea *retry-job* o *retry-current-job*.
El valor predeterminado es ‘30’.

**entero-no-negativo job-retry-limit** [parámetro de cups-configuration]
Especifica el número de reintentos que se llevan a cabo con los trabajos. De manera habitual se usa con colas de fax pero también puede usarse con colas de impresión normal cuya política de error sea *retry-job* o *retry-current-job*.
El valor predeterminado es ‘5’.
 Capítulo 8: Configuración del sistema 231

boolean keep-alive? [parámetro de cups-configuration]
Especifica si se permiten conexiones “keep-alive” de HTTP.
El valor predeterminado es ‘#t’

entero-no-negativo keep-alive-timeout [parámetro de cups-configuration]
Especifica por cuanto tiempo permanece abierta una conexión de cliente en estado de espera, en segundos.
El valor predeterminado es ‘30’.

entero-no-negativo limit-request-body [parámetro de cups-configuration]
Especifica el tamaño máximo de los ficheros de impresión, peticiones IPP y datos de formularios HTTP. Un límite de 0 desactiva la comprobación del límite.
El valor predeterminado es ‘0’.

lista-cadenas-multilínea listen [parámetro de cups-configuration]
Escucha a la espera de conexiones en las interfaces especificadas. Se aceptan valores con la forma dirección:puerto, donde dirección es o bien una dirección IPv6 entre corchetes, una dirección IPv4 o * para indicar todas las direcciones. Los valores también pueden ser nombres de fichero de sockets de dominio de UNIX locales. La directiva “Listen” es similar a la directiva “Port”, pero le permite la restricción del acceso a interfaces o redes específicas.

entero-no-negativo listen-back-log [parámetro de cups-configuration]
Especifica el número de conexiones pendientes permitidas. Habitualmente afecta de manera exclusiva a servidores con mucha carga de trabajo que han alcanzado el límite de clientes “MaxClients”, pero también puede ser provocado por grandes números de conexiones simultáneas. Cuando se alcanza el límite, el sistema operativo rechaza conexiones adicionales hasta que el planificador pueda aceptar las pendientes.
El valor predeterminado es ‘128’.

lista-location-access-control [parámetro de cups-configuration]
location-access-controls
Especifica un conjunto adicional de controles de acceso.
Los campos disponibles de location-access-controls son:

nombre-fichero path [parámetro de location-access-controls]
Especifica la ruta URI sobre la que el control de acceso tendrá efecto.

lista-access-control [parámetro de location-access-controls]
access-controls
Controles de acceso para todos los accesos a esta ruta, en el mismo formato que access-controls de operation-access-control.
El valor predeterminado es ‘()’.

lista-method-access-control [parámetro de location-access-controls]
method-access-controls
Controles de acceso para accesos con métodos específicos para esta ruta.
El valor predeterminado es ‘()’.
Los campos disponibles de method-access-controls son:
boolean reverse?  [parámetro de method-access-controls]
   Si es #t, los controles de acceso son efectivos con todos los métodos excepto los métodos listados. En otro caso, son efectivos únicamente con los métodos listados.
   El valor predeterminado es ‘#f’

lista-métodos methods  [parámetro de method-access-controls]
   Métodos con los cuales este control de acceso es efectivo.
   El valor predeterminado es ‘()’.

lista-control-acceso  [parámetro de method-access-controls]
   access-controls
   Directivas de control de acceso, como una lista de cadenas. Cada cadena debe ser una directiva, como "Order allow,deny".
   El valor predeterminado es ‘()’.

entero-no-negativo log-debug-history  [parámetro de cups-configuration]
   Especifica el número de mensajes de depuración que se retienen para el registro si sucede un error en un trabajo de impresión. Los mensajes de depuración se registran independientemente de la configuración de “LogLevel”.
   El valor predeterminado es ‘100’.

nivel-registro log-level  [parámetro de cups-configuration]
   Especifica el nivel de depuración del fichero “ErrorLog”. El valor none inhibe todos los registros mientras que debug2 registra todo.
   El valor predeterminado es ‘info’

formato-tiempo-registro  [parámetro de cups-configuration]
   log-time-format
   Especifica el formato de la fecha y el tiempo en los ficheros de registro. El valor standard registra con segundos completos mientras que usecs registra con microsegundos.
   El valor predeterminado es ‘standard’.

entero-no-negativo max-clients  [parámetro de cups-configuration]
   Especifica el número de clientes simultáneos máximo que son admitidos por el planificador.
   El valor predeterminado es ‘100’.

entero-no-negativo max-clients-per-host  [parámetro de cups-configuration]
   Especifica el número de clientes simultáneos máximo que se permiten desde una única dirección.
   El valor predeterminado es ‘100’.

entero-no-negativo max-copies  [parámetro de cups-configuration]
   Especifica el número de copias máximo que una usuaria puede imprimir con cada trabajo.
   El valor predeterminado es ‘9999’.
**Capítulo 8: Configuración del sistema**

233

**entero-no-negativo max-hold-time** [parámetro de cups-configuration]

Especifica el tiempo máximo que un trabajo puede permanecer en el estado de espera indefinite antes de su cancelación. Un valor de 0 desactiva la cancelación de trabajos en espera.

El valor predeterminado es ‘0’.

**entero-no-negativo max-jobs** [parámetro de cups-configuration]

Especifica el número de trabajos simultáneos máximo permitido. El valor 0 permite un número ilimitado de trabajos.

El valor predeterminado es ‘500’.

**entero-no-negativo max-jobs-per-printer** [parámetro de cups-configuration]

Especifica el número de trabajos simultáneos máximo que se permite por impresora. Un valor de 0 permite hasta “MaxJobs” por impresora.

El valor predeterminado es ‘0’.

**entero-no-negativo max-jobs-per-user** [parámetro de cups-configuration]

Especifica el número de trabajos simultáneos máximo que se permite por usuaria. Un valor de 0 permite hasta “MaxJobs” por usuaria.

El valor predeterminado es ‘0’.

**entero-no-negativo max-job-time** [parámetro de cups-configuration]

Especifica el tiempo de duración de la impresión máximo que un trabajo puede tomar antes de ser cancelado, en segundos. El valor 0 desactiva la cancelación de trabajos “atascados”\(^{10}\).

El valor predeterminado es ‘10800’.

**entero-no-negativo max-log-size** [parámetro de cups-configuration]

Especifica el tamaño máximo de los ficheros de registro antes de su rotación, en bytes. El valor 0 desactiva la rotación de registros.

El valor predeterminado es ‘1048576’.

**entero-no-negativo multiple-operation-timeout** [parámetro de cups-configuration]

Especifica el tiempo máximo permitido entre ficheros en un trabajo de impresión con múltiples ficheros, en segundos.

El valor predeterminado es ‘300’.

**string page-log-format** [parámetro de cups-configuration]

Especifica el formato de las líneas PageLog. Las secuencias de caracteres que comienzan con el signo de porcentaje (’%’) se reemplazan con la información correspondiente, mientras que el resto de caracteres se copia de manera literal. Se reconocen las siguientes secuencias:

\[%%\] inserta literalmente un símbolo de porcentaje

\[10\] El estado es ‘stuck’ en inglés.
inserta el valor del atributo IPP especificado
‘%C’ inserta el número de copias para la página actual
‘%P’ inserta el número de página actual
‘%T’ inserta la fecha y hora actuales en el formato común de registro
‘%j’ introduce el ID del trabajo
‘%p’ inserta el nombre de impresora
‘%u’ inserta el nombre de usuario
La cadena vacía desactiva el registro de página. La cadena %p %u %j %T %P
%C %{job-billing} %{job-originating-host-name} %{job-name} %{media}
%{sides} crea un registro de página con los elementos estándar.
El valor predeterminado es ‘""’.

variables-entorno [parámetro de cups-configuration]
environment-variables
Proporciona la o las variables de entorno especificadas a los procesos iniciados; una
lista de cadenas.
El valor predeterminado es ‘()’.

lista-policy-configuration policies [parámetro de cups-configuration]
Especifica las políticas de control de acceso con nombre.
Los campos disponibles de policy-configuration son:

string name [parámetro de policy-configuration]
El nombre de la política.

string job-private-access [parámetro de policy-configuration]
Especifica una lista de acceso para los valores privados de un trabajo.
@ACL se sustituye con los valores “requesting-user-name-allowed” o
“requesting-user-name-denied” de la impresora. @OWNER se sustituye con la
propietaria del trabajo. @SYSTEM se sustituye con los grupos enumerados en
el campo system-group de la configuración files-config, que se traduce
en el fichero cups-files.conf(5). Otros elementos de configuración de la
lista de acceso posibles incluyen nombres de usuario específicos y @group para
indicar miembros de un grupo específico. La lista de acceso también puede
simplemente ser all o default.
El valor predeterminado es ‘"@OWNER @SYSTEM"’.

string job-private-values [parámetro de policy-configuration]
Especifica la lista de valores de trabajos a hacer privados, o bien all, default, o
none.
El valor predeterminado es ‘"job-name job-originating-host-name
job-originating-user-name phone"’.
string [parámetro de policy-configuration]
subscription-private-access
Especifica una lista de acceso para los valores privados de una subscripción. @ACL se sustituye con los valores “requesting-user-name-allowed” o “requesting-user-name-denied” de la impresora. @OWNER se sustituye con la propietaria del trabajo. @SYSTEM se sustituye con los grupos enumerados en el campo system-group de la configuración files-config, que se traduce en el fichero cups-files.conf(5). Otros elementos de configuración de la lista de acceso posibles incluyen nombres de usuaria específicos y @group para indicar miembros de un grupo específico. La lista de acceso también puede simplemente ser all o default.
El valor predeterminado es ‘"@OWNER @SYSTEM"’.

string [parámetro de policy-configuration]
subscription-private-values
Especifica la lista de valores de trabajos a hacer privados, o bien all, default, o none.
El valor predeterminado es ‘"notify-events notify-pull-method notify-recipient-uri notify-subscriber-user-name notify-user-data"’

lista-operation-access-control [parámetro de policy-configuration]
access-controls
Control de acceso para operaciones de IPP.
El valor predeterminado es ‘( )’.

boolean-o-entero-no-negativo [parámetro de cups-configuration]
preserve-job-files
Especifica si los ficheros del trabajo (documentos) se preservan tras la impresión de un trabajo. Si se especifica un valor numérico, los ficheros del trabajo se preservan durante el número indicado de segundos tras la impresión. En otro caso, el valor booleano determina la conservación de manera indefinida.
El valor predeterminado es ‘86400’.

boolean-o-entero-no-negativo [parámetro de cups-configuration]
preserve-job-history
Especifica si la historia del trabajo se preserva tras la impresión de un trabajo. Si se especifica un valor numérico, la historia del trabajo se conserva tras la impresión el número de segundos indicado. Si es #t, la historia del trabajo se conserva hasta que se alcance el límite de trabajos “MaxJobs”.
El valor predeterminado es ‘#t’

entero-no-negativo reload-timeout [parámetro de cups-configuration]
Especifica el tiempo a esperar hasta la finalización del trabajo antes de reiniciar el planificador.
El valor predeterminado es ‘30’.
string rip-cache  
Especifica la cantidad máxima de memoria usada durante la conversión de documentos en imágenes para una impresora.
El valor predeterminado es "128m".

string server-admin  
Especifica la dirección de correo electrónico de la administradora del servidor.
El valor predeterminado es "root@localhost.localdomain".

lista-nombres-máquina-o-** server-alias  
La directiva ServerAlias se usa para la validación de la cabecera HTTP Host cuando los clientes se conecten al planificador desde interfaces externas. El uso del nombre especial * puede exponer su sistema a ataques basados en el navegador web de reenlazado DNS ya conocidos, incluso cuando se accede a páginas a través de un cortafuegos. Si el descubrimiento automático de nombres alternativos no funcionase, le recomendamos enumerar cada nombre alternativo con una directiva ServerAlias en vez del uso de *
El valor predeterminado es *.

string server-name  
Especifica el nombre de máquina completamente cualificado del servidor.
El valor predeterminado es "localhost".

server-tokens server-tokens  
Especifica qué información se incluye en la cabecera Server de las respuestas HTTP. None desactiva la cabecera Server. ProductOnly proporciona CUPS. Major proporciona CUPS 2. Minor proporciona CUPS 2.0. Minimap proporciona CUPS 2.0.0. OS proporciona CUPS 2.0.0 (uname) donde uname es la salida de la orden uname. Full proporciona CUPS 2.0.0 (uname) IPP/2.0.
El valor predeterminado es 'Minimal'.

lista-cadenas-multilínea ssl-listen  
Escucha en las interfaces especificadas a la espera de conexiones cifradas. Se aceptan valores con la forma dirección:puerto, siendo dirección o bien una dirección IPv6 entre corchetes, o bien una dirección IPv4, o bien * que representa todas las direcciones.
El valor predeterminado es ‘()’.

opciones-ssl ssl-options  
Determina las opciones de cifrado. De manera predeterminada, CUPS permite únicamente el cifrado mediante TLS v1.0 o superior mediante el uso de modelos de cifrado de conocida seguridad. La seguridad se reduce cuando se usan opciones Allow y se aumenta cuando se usan opciones Deny. La opción AllowRC4 permite el cifrado RC4 de 128 bits, necesario para algunos clientes antiguos que no implementan los modelos más modernos. La opción AllowSSL3 desactiva SSL v3.0, necesario para algunos clientes antiguos que no implementan TLS v1.0. La opción DenyCBC desactiva todos los modelos de cifrado CBC. La opción DenyTLS1.0 desactiva TLS v1.0—esto fuerza la versión mínima del protocolo a TLS v1.1.
El valor predeterminado es ‘()’.
Capítulo 8: Configuración del sistema

boolean strict-conformance? [parámetro de cups-configuration]
   Especifica si el planificador exige que los clientes se adhieran de manera estricta a las especificaciones IPP.
   El valor predeterminado es ‘#f’

entero-no-negativo timeout [parámetro de cups-configuration]
   Especifica el plazo de las peticiones HTTP, en segundos.
   El valor predeterminado es ‘300’.

boolean web-interface? [parámetro de cups-configuration]
   Especifica si se debe activar la interfaz web.
   El valor predeterminado es ‘#f’

En este punto probablemente esté pensando, “querido manual de Guix, me gusta todo esto, pero... ¿cuando se acaban las opciones de configuración?!”. De hecho ya terminan. No obstante, hay un punto más: puede ser que ya tenga un fichero cupsd.conf que desee usar. En ese caso, puede proporcionar un objeto opaque-cups-configuration como la configuración de cups-service-type.

Los campos disponibles de opaque-cups-configuration son:

paquete cups [parámetro de opaque-cups-configuration]
   El paquete CUPS.

string cupsd.conf [parámetro de opaque-cups-configuration]
   El contenido de cupsd.conf, como una cadena.

string cups-files.conf [parámetro de opaque-cups-configuration]
   El contenido del fichero cups-files.conf, como una cadena.

Por ejemplo, si el contenido de sus ficheros cupsd.conf y cups-files.conf estuviese en cadenas del mismo nombre, podría instanciar un servicio CUPS de esta manera:

(service cups-service-type
   (opaque-cups-configuration
      (cupsd.conf cupsd.conf)
      (cups-files.conf cups-files.conf)))

8.8.7 Servicios de escritorio

El módulo (gnu services desktop) proporciona servicios que son útiles habitualmente en el contexto de una configuración de “escritorio”—es decir, en una máquina que execute un servidor gráfico, posiblemente con interfaces gráficas, etcétera. También define servicios que proporcionan entornos de escritorio específicos como GNOME, Xfce o MATE.

Para simplificar las cosas, el módulo define una variable que contiene el conjunto de servicios que las usuarias esperarían de manera habitual junto a un entorno gráfico y de red:

%desktop-services [Variable Scheme]
   Es una lista de servicios que se construye en base a %base-services y añade o ajusta servicios para una configuración de “escritorio” típica.
En particular, añade un gestor de ingreso al sistema gráfico (véase Sección 8.8.5 [Sistema X Window], página 218), herramientas para el bloqueo de la pantalla, una herramienta de gestión de redes (véase Sección 8.8.4 [Servicios de red], página 200) y gestión de modem (véase Sección 8.8.4 [Servicios de red], página 200), servicios de gestión de la energía y el color, el gestor de asientos e ingresos al sistema elogind, el servicio de privilegios Polkit, el servicio de geolocalización GeoClue, el daemon AccountsService que permite a las usuarias autorizadas el cambio de contraseñas del sistema, un cliente NTP (véase Sección 8.8.4 [Servicios de red], página 200), el daemon Avahi y configura el servicio del selector de servicios de nombres para que pueda usar nss-mdns (véase Sección 8.11 [Selector de servicios de nombres], página 391).

La variable %desktop-services puede usarse como el campo services de una declaración operating-system (véase Sección 8.2 [Referencia de operating-system], página 168).

Además, el gnome-desktop-service-type, xfce-desktop-service, mate-desktop-service-type and enlightenment-desktop-service-type procedimientos pueden añadir GNOME, Xfce, MATE y/o Enlightenment a un sistema. To “add GNOME” significa que servicios de nivel de sistema tales como los ayudantes de ajuste de brillo y las utilidades de administración de energía son añadidos al sistema, extendiendo polkit y dbus de forma apropiada, permitiendo que GNOME opere con privilegios elevados en un número limitado de interfaces de sistema de propósito especial. Además, añadir un servicio hecho por gnome-desktop-service-type añade el paquete metapackage GNOME al perfil del sistema, y extiende polkit con las acciones de gnome-settings-daemon.

Los entornos de escritorio en Guix usan el servidor gráfico Xorg de manera predeterminada. Si desea usar el protocolo de servidor gráfico más nuevo llamado Wayland, debe usar el servicio sddm-service en vez de GDM como gestor gráfico de ingreso al sistema. Una vez hecho, debe seleccionar la sesión “GNOME (Wayland)” en SDDM. Alternativamente, puede intentar iniciar GNOME en Wayland de manera manual desde una TTY con la orden “XDG_SESSION_TYPE=wayland exec dbus-run-session gnome-session”. Actualmente únicamente GNOME tiene implementación para Wayland.

gnome-desktop-service-type [Variable Scheme]

Es el tipo del servicio que añade el entorno de escritorio GNOME (https://www.gnome.org). Su valor es un objeto gnome-desktop-configuration (véase a continuación).

Este servicio añade el paquete gnome al perfil del sistema, y extiende polkit con las acciones de gnome-settings-daemon.

gnome-desktop-configuration [Tipo de datos]

Registro de configuración para el entorno de escritorio GNOME.
gnome (predeterminado: gnome)
El paquete GNOME usado.

xfce-desktop-service-type
[Variable Scheme]
Este es el tipo de un servicio para ejecutar el entorno de escritorio https://xfce.org (Xfce). Su valor es un objeto xfce-desktop-configuration (véase a continuación).
Este servicio añade el paquete xfce al perfil del sistema, y extiende polkit con la capacidad de thunar para manipular el sistema de ficheros como root dentro de una sesión de usuario, tras la identificación de la usuario con la contraseña de administración.

xfce-desktop-configuration
[Tipo de datos]
Registro de configuración para el entorno de escritorio Xfce.
xfce (predeterminado: xfce)
El paquete Xfce usado.

mate-desktop-service-type
[Variable Scheme]
Es el tipo del servicio que ejecuta el entorno de escritorio MATE (https://mate-desktop.org/). Su valor es un objeto mate-desktop-configuration (véase a continuación).
Este servicio añade el paquete mate al perfil del sistema, y extiende polkit con acciones de mate-settings-daemon.

mate-desktop-configuration
[Tipo de datos]
Registro de configuración para el entorno de escritorio MATE.
mate (predeterminado: mate)
El paquete MATE usado.

enlightenment-desktop-service-type
[Variable Scheme]
Devuelve un servicio que añade el paquete enlightenment al perfil del sistema, y extiende dbus con acciones de efl.
enlightenment-desktop-service-configuration
[Tipo de datos]
enlightenment (predeterminado: enlightenment)
El paquete enlightenment usado.

Debido a que los servicios de escritorio GNOME, Xfce y MATE incorporan tantos paquetes, la variable %desktop-services no incluye ninguno de manera predeterminada. Para añadir GNOME, Xfce o MATE, simplemente use cons junto a %desktop-services en el campo services de su declaración operating-system:

(use-modules (gnu))
(use-service-modules desktop)
(operating-system
  ...
  ;; cons* añade elementos a la lista proporcionada en el último
  ;; parámetro.
  (services (cons* (service gnome-desktop-service-type)
                  (service xfce-desktop-service)
Una vez realizado, estos entornos de escritorio se encontrarán como opciones disponibles en la ventana del gestor gráfico de ingreso al sistema.

Las definiciones de servicio incluidas realmente en %desktop-services y proporcionadas por (gnu services dbus) y (gnu services desktop) se describen a continuación.

`dbus-service [#:dbus dbus] [#:services ()]`  
[Procedimiento Scheme]  
Devuelve un servicio que ejecuta el “bus del sistema”, mediante el uso de dbus, con configuración para services.

D-Bus (https://dbus.freedesktop.org/) es una herramienta para la facilitación de intercomunicación entre procesos. Su bus del sistema se usa para permitir la comunicación entre y la notificación de eventos a nivel de sistema a los servicios.

`services` debe ser una lista de paquetes que proporcionen un directorio `etc/dbus-1/system.d` que contenga ficheros de configuración y políticas adicionales de D-Bus. Por ejemplo, para permitir a avahi-daemon el uso del bus del sistema, `services` debe tener el valor (list avahi).

`elogind-service [#:config config]`  
[Procedimiento Scheme]  
Devuelve un servicio que ejecuta el daemon de gestión de ingreso al sistema y de asientos elogind. Elogind (https://github.com/elogind/elogind) expone una interfaz D-Bus que puede usarse para conocer las usuarias que han ingresado en el sistema, conocer qué tipo de sesiones tienen abiertas, suspender el sistema o inhibir su suspensión, reiniciar el sistema y otras tareas.

Elogind maneja la mayor parte de los eventos a nivel de sistema de alimentación de su máquina, por ejemplo mediante la suspensión del sistema cuando se cierre la tapa, o mediante el apagado al pulsar la tecla correspondiente.

El parámetro `config` especifica la configuración de elogind, y debería resultar en una invocación (elogind-configuration (parámetro valor)...). Los parámetros disponibles y sus valores predeterminados son:

```
kill-user-processes?
  #f
kill-only-users
  ()
kill-exclude-users
  ("root")
inhibit-delay-max-seconds
  5
handle-power-key
  poweroff
handle-suspend-key
  suspend
handle-hibernate-key
  hibernate
```
handle-lid-switch
  suspend
handle-lid-switch-docked
  ignore
power-key-ignore-inhibited?
  #f
suspend-key-ignore-inhibited?
  #f
hibernate-key-ignore-inhibited?
  #f
lid-switch-ignore-inhibited?
  #t
holdoff-timeout-seconds
  30
idle-action
  ignore
idle-action-seconds
  (* 30 60)
runtime-directory-size-percent
  10
runtime-directory-size
  #f
remove-ipc?
  #t
suspend-state
  ("mem" "standby" "freeze")
suspend-mode
  ()
hibernate-state
  ("disk")
hibernate-mode
  ("platform" "shutdown")
hybrid-sleep-state
  ("disk")
hybrid-sleep-mode
  ("suspend" "platform" "shutdown")

accountsservice-service [#:accountsservice]

Devuelve un servicio que ejecuta AccountsService, un servicio del sistema para la enumeración de cuentas disponibles, el cambio de sus contraseñas, etcétera. AccountsService se integra con PolicyKit para permitir a las usuarias sin privilegios...
la adquisición de la capacidad de modificar la configuración de su sistema. Véase la página web de AccountsService (https://www.freedesktop.org/wiki/Software/AccountsService/) para más información.

El parámetro accountsservice es el paquete accountsservice que se expondrá como un servicio.

polkit-service [#:polkit polkit] [Procedimiento Scheme]
Devuelve un servicio que ejecuta el servicio de gestión de privilegios Polkit (https://www.freedesktop.org/wiki/Software/polkit/), que permite a las administradoras del sistema la concesión de permisos sobre operaciones privilegiadas de manera estructurada. Mediante las consultas al servicio Polkit, un componente del sistema con privilegios puede conocer cuando debe conceder capacidades adicionales a usuarias ordinarias. Por ejemplo, se le puede conceder la capacidad a una usuaria ordinaria de suspender el sistema si la usuaria ingresó de forma local.

polkit-wheel-service [Variable Scheme]
Servicio que añade a las usuarias del grupo wheel como administradoras del servicio Polkit. Esto hace que se solicite su propia contraseña a las usuarias del grupo wheel cuando realicen acciones administrativas en vez de la contraseña de root, de manera similar al comportamiento de sudo.

upower-service-type [Variable Scheme]
Servicio que ejecuta https://upower.freedesktop.org/, upowerd, un monitor a nivel de sistema de consumo de energía y niveles de batería, con las opciones de configuración proporcionadas.

Implementa la interfaz D-Bus org.freedesktop.UPower, y se usa de forma notable en GNOME.

upower-configuration [Tipo de datos]
Tipo de datos que representa la configuración de UPower.

upower (predeterminado: upower)
Paquete usado para upower.

watts-up-pro? (predeterminado: #f)
Permite el uso del dispositivo Watts Up Pro.

poll-batteries? (predeterminado: #t)
Usa el servicio de consulta del núcleo para los cambios en niveles de batería.

ignore-lid? (predeterminado: #f)
Ignora el estado de la tapa, puede ser útil en caso de ser incorrecto un dispositivo determinado.

use-percentage-for-policy? (predeterminado: #f)
Determina si se debe usar la política basada en el porcentaje de batería. La opción predeterminada es el uso del tiempo restante, cambie a #t para usar el porcentaje.
Capítulo 8: Configuración del sistema

percentage-low (predeterminado: 10)
  Cuando use-percentage-for-policy? es #t, determina el porcentaje en el que la carga de la batería se considera baja.

percentage-critical (predeterminado: 3)
  Cuando use-percentage-for-policy? es #t, determina el porcentaje en el que la carga de la batería se considera crítica.

percentage-action (predeterminado: 2)
  Cuando use-percentage-for-policy? es #t, determina el porcentaje en el que se tomará la acción.

time-low (predeterminado: 1200)
  Cuando use-percentage-for-policy? es #t, determina el tiempo restante en segundos con el que carga de la batería se considera baja.

time-critical (predeterminado: 300)
  Cuando use-percentage-for-policy? es #t, determina el tiempo restante en segundos con el que carga de la batería se considera crítica.

time-action (predeterminado: 120)
  Cuando use-percentage-for-policy? es #t, determina el tiempo restante en segundos con el que se tomará la acción.

critical-power-action (predeterminada: ‘hybrid-sleep)
  La acción tomada cuando se alcanza percentage-action o time-action (dependiendo de la configuración de use-percentage-for-policy?).
  Los valores posibles son:
  • ‘power-off
  • ‘hibernate
  • ‘hybrid-sleep.

udisks-service [#::udisks udisks]
  [Procedimiento Scheme]
  Devuelve un servicio para UDisks (https://udisks.freedesktop.org/docs/latest/), un daemon de gestión de discos que proporciona interfaces de usuaria con notificaciones y formas de montar/desmontar discos. Los programas que se comunican con UDisk incluyen la orden udisksctl, parte de UDIsks, y la utilidad “Discos” de GNOME.

colord-service-type
  [Variable Scheme]
  Devuelve un servicio que ejecuta colord, un servicio del sistema con una interfaz D-Bus para la gestión de perfiles de dispositivos de entrada y salida como la pantalla y el escáner. Se usa de forma notable por parte de la herramienta gráfica de “Gestión de color” de GNOME. Véase la página web de colord (https://www.freedesktop.org/software/colord/) para más información.

gеoclue-application-name [#::allowed? #t]
  [#::system? #f] [#::users ’()]
  [Procedimiento Scheme]
  Devuelve una configuración que permite a una aplicación el acceso a los datos de posicionamiento de GeoClue. nombre es el Desktop ID de la aplicación, sin la parte
Capítulo 8: Configuración del sistema

... .desktop. Si el valor de allowed? es verdadero, la aplicación tendrá acceso a la información de posicionamiento de manera predeterminada. El valor booleano system? indica si una aplicación es un componente de sistema o no. Por último, users es una lista de UID de todas las usuarias para las que esta aplicación tiene permitido el acceso de información. Una lista de usuarias vacía significa que se permiten todas las usuarias.

sane-service-type

[Procedimiento Scheme]
Este servicio proporciona acceso a escáner a través de SANE (http://www.sane-project.org) instalando las reglas de udev necesarias.

%standard-geoclue-applications

[Variable Scheme]
La lista estándar de configuraciones de GeoClue de aplicaciones conocidas, proporcionando autoridad a la utilidad de fecha y hora de GNOME para obtener la localización actual para ajustar la zona horaria, y permitiendo que los navegadores Icecat y Epiphany puedan solicitar información de localización. Tanto IceCat como Epiphany solicitan permiso a la usuaria antes de permitir a una página web conocer la ubicación de la usuaria.

google-slicer-service

[Procedimiento Scheme]
Devuelve un servicio que ejecuta el servicio de posicionamiento GeoClue. Proporciona una interfaz D-Bus que permite a las aplicaciones solicitar una posición física de la usuaria, y de manera opcional añadir información a bases de datos de posicionamiento en línea. Véase la página web de GeoClue (https://wiki.freedesktop.org/www/Software/GeoClue/) para obtener más información.

bluetooth-service

[Procedimiento Scheme]
Devuelve un servicio que ejecuta el daemon bluetoothd que gestiona todos los dispositivos Bluetooth y proporciona cierto número de interfaces D-Bus. Cuando auto-enable? es verdadero, el controlador bluetooth tendrá alimentación automáticamente tras el arranque, lo que puede ser útil si se posee un teclado o un ratón bluetooth.

Las usuarias necesitan la pertenencia al grupo lp para el acceso al servicio D-Bus.

gnome-keyring-service-type

[Variable Scheme]
Es el tipo del servicio que añade el entorno de escritorio el anillo de claves de GNOME (https://wiki.gnome.org/Projects/GnomeKeyring). Su valor es un objeto gnome-keyring-configuration (véase a continuación).

Este servicio añade el paquete gnome-keyring al perfil del sistema y extiende PAM con entradas que usan pam_gnome_keyring.so, las cuales desbloquean el anillo de claves del sistema de la usuaria cuando ingrese en el sistema o cambie su contraseña con passwd.
Capítulo 8: Configuración del sistema

gnome-keyring-configuration [Tipo de datos]
Registro de configuración para el servicio del anillo de claves de GNOME.

keyring (predeterminado: gnome-keyring)
El paquete GNOME keyring usado.

pam-services
Una lista de pares (servicio . tipo) que denotan los servicios de PAM que deben extenderse, donde servicio es el nombre de un servicio existente que debe extenderse y tipo es login o passwd.
Si se proporciona login, añade un campo opcional pam_gnome_keyring.so al bloque de identificación sin parámetros y al bloque de sesión con auto_start. Si se proporciona passwd, añade un campo opcional pam_gnome_keyring.so al bloque de contraseña sin parámetros.
De manera predeterminada, este campo contiene “gdm-password” con el valor login y “passwd” tiene valor passwd.

8.8.8 Servicios de sonido
El módulo (gnu services sound) proporciona un servicio para la configuración del sistema ALSA (arquitectura avanzada de sonido de Linux), el cual establece PulseAudio como el controlador de ALSA preferido para salida de sonido.

alsa-service-type [Variable Scheme]
Es el tipo para el sistema ALSA (https://alsa-project.org/) (Arquitectura de sonido avanzada de Linux), que genera el fichero de configuración /etc/asound.conf. El valor para este tipo es un registro alsa-configuration como en el ejemplo:

(service alsa-service-type)
Véase a continuación más detalles sobre alsa-configuration.

alsa-configuration [Tipo de datos]
Tipo de datos que representa la configuración para alsa-service.

alsa-plugins (predeterminados: alsa-plugins)
El paquete alsa-plugins usado.

pulseaudio? (predeterminado: #t)
Whether ALSA applications should transparently be made to use the PulseAudio (https://www.pulseaudio.org/) sound server.
El uso de PulseAudio le permite la ejecución de varias aplicaciones que produzcan sonido al mismo tiempo y su control individual mediante pavucontrol, entre otras opciones.

extra-options (predeterminado: "")
Cadena a añadir al final del fichero /etc/asound.conf.

Las usuarias individuales que deseen forzar la configuración de ALSA en el sistema para sus cuentas pueden hacerlo con el fichero ~/.asoundrc:

# En guix tenemos que especificar la ruta absoluta del módulo.
pcm_type.jack {
  lib "/home/alicia/.guix-profile/lib/alsa-lib/libasound_module_pcm_jack.so"
}

# Redirección de ALSA a jack:
# <http://jackaudio.org/faq/routing_alsa.html>.
pcm.rawjack {
  type jack
  playback_ports {
    0 system:playback_1
    1 system:playback_2
  }
  capture_ports {
    0 system:capture_1
    1 system:capture_2
  }
}

pcm.!default {
  type plug
  slave {
    pcm "rawjack"
  }
}


pulseaudio-service-type

This is the type for the PulseAudio (https://www.pulseaudio.org/) sound server. It exists to allow system overrides of the default settings via pulseaudio-configuration, see below.

Aviso: Este servicio no asegura en sí que el paquete pulseaudio exista en su máquina. Únicamente añade los ficheros de configuración, como se detalla a continuación. En el caso (ciertamente poco probable), de que se encuentre si un paquete pulseaudio pulseaudio, considere activarlo a través del tipo alsa-service-type mostrado previamente.

pulseaudio-configuration

Tipo de datos que representa la configuración para pulseaudio-service.

client-conf (predeterminada: '() )
Lista de opciones de configuración de client.conf. Acepta una lista de cadenas o pares símbolo-valor. Las cadenas se introducirán literalmente añadiendo un salto de línea. Los pares tendrán formato “clave = valor”, de nuevo añadiendo un salto de línea.
Capítulo 8: Configuración del sistema

**daemon-conf** (predeterminada: `'(flat-volumes . no))

Lista de opciones de configuración de `daemon.conf`, con el mismo formato que `client-conf`.

**script-file** (predeterminado: `(file-append pulseaudio 
"/etc/pulse/default.pa")`)

Fichero del guión usado como `default.pa`

**system-script-file** (predeterminado: `(file-append pulseaudio 
"/etc/pulse/system.pa")`)

Fichero del guión usado como `system.pa`

**ladspa-service-type** [Variable Scheme]

Este servicio proporciona valor a la variable `LADSPA_PATH`, de manera que los programas que lo tengan en cuenta, por ejemplo PulseAudio, puedan cargar módulos LADSPA.

El siguiente ejemplo configura el servicio para permitir la activación de los módulos del paquete `swh-plugins`:

```scheme
(service ladspa-service-type
   (ladspa-configuration (plugins (list swh-plugins))))
```


### 8.8.9 Servicios de bases de datos

El módulo (**gnu services databases**) proporciona los siguientes servicios.

**postgresql-service** [#postgresql postgresql] [Procedimiento Scheme]

`[#:config-file] [#:data-directory “/var/lib/postgresql/data”] [#:port 5432] [#:locale “en_US.utf8”] [#:extension-packages ’()]` Devuelve un servicio que ejecuta `postgresql`, el servidor de bases de datos PostgreSQL.

El daemon de PostgreSQL carga su configuración de tiempo de ejecución desde `config-file`, crea un cluster de bases de datos con `locale` como la localización predeterminada, almacenado en `data-directory`. Una vez hecho esto, escucha en `port`.

Las extensiones adicionales se cargan de paquetes enumerados en `extension-packages`. Las extensiones están disponibles en tiempo de ejecución. Por ejemplo, para crear una base de datos geográfica con la extensión `postgis`, una usuaria podría configurar el servicio `postgresql-service` como en este ejemplo:

```scheme
(use-package-modules databases geo)

(operating-system ...

   ;; postgresql es necesario para ejecutar ‘psql’ pero no se necesita
   ;; postgis para un funcionamiento correcto.
   (packages (cons* postgresql %base-packages))
   (services
      (cons*
         (postgresql-service #:extension-packages (list postgis))
```
Una vez hecho, la extensión estará visible y podrá inicializar una base de datos geográfica de este modo:

```
psql -U postgres
> create database pruebapostgis;
> \connect pruebapostgis;
> create extension postgis;
> create extension postgis_topology;
```

No es necesaria la adición de este campo para extensiones incluidas en la distribución oficial\(^1\) como hstore o dblink, puesto que ya pueden cargarse en postgresql. Este campo únicamente es necesario para extensiones proporcionadas por otros paquetes.

**mysql-service** [#:config (mysql-configuration)]

[Procedimiento Scheme]

Devuelve un servicio que ejecuta mysqld, el servidor de base de datos MySQL o MariaDB.

El parámetro opcional `config` especifica la configuración para `mysqld`, que debe ser un objeto `<mysql-configuration>`.

**mysql-configuration** [Tipo de datos]

Tipo de datos que representa la configuración para `mysql-service`.

- **mysql** (predeterminado: `mariadb`)
  - Objeto de paquete del servidor de bases de datos MySQL, puede ser tanto `mariadb` como `mysql`.
  - Para MySQL, se mostrará una contraseña de root temporal durante el tiempo de activación. Para MariaDB, la contraseña de root está vacía.

- **port** (predeterminado: 3306)
  - Puerto TCP en el que escucha el servidor de bases de datos a la espera de conexiones entrantes.

**memcached-service-type** [Variable Scheme]

Este es el tipo de servicio para el servicio Memcached ([https://memcached.org/](https://memcached.org/)), que proporciona caché distribuida en memoria. El valor para este tipo de servicio es un objeto `memcached-configuration`.

**(service memcached-service-type)**

**memcached-configuration** [Tipo de datos]

Tipo de datos que representa la configuración de memcached.

- **memcached** (predeterminado: `memcached`)
  - El paquete de Memcached usado.

- **interfaces** (predeterminadas: `("0.0.0.0")`)
  - Interfaces de red por las que se esperan conexiones.

- **tcp-port** (predeterminado: 11211)
  - Puerto en el que se deben aceptar conexiones.

---

\(^{1}\) NdT: “contrib” de “contributed” en inglés, “contribuciones” podría entenderse en castellano.
Capítulo 8: Configuración del sistema

udp-port (predeterminado: 11211)
Puerto en el que se deben aceptar conexiones UDP, el valor 0 desactiva la escucha en un socket UDP.

additional-options (predeterminadas: '()')
Opciones de línea de órdenes adicionales que se le proporcionarán a memcached.

mongodb-service-type [Variable Scheme]

(service mongodb-service-type)

mongodb-configuration [Tipo de datos]
Tipo de datos que representa la configuración de GPM.

mongodb (predeterminado: mongodb)
El paquete MongoDB usado.

config-file (predeterminado: %default-mongodb-configuration-file)
El fichero de configuración de MongoDB.

data-directory (predeterminado: "/var/lib/mongodb")
Este valor se usa para la creación del directorio, de modo que exista y sea propiedad de la usuaria mongodb. Debe corresponder con el directorio de datos para el que se ha configurado MongoDB a través del fichero de configuración.

redis-service-type [Variable Scheme]
Es el tipo de servicio para el almacén de clave/valor Redis (https://redis.io/), cuyo valor es un objeto redis-configuration.

redis-configuration [Tipo de datos]
Tipo de datos que representa la configuración de redis.

redis (predeterminado: redis)
El paquete Redis usado.

bind (predeterminada: "127.0.0.1")
La interfaz de red en la que se escucha.

port (predeterminado: 6379)
Puerto en el que se aceptan conexiones, el valor 0 desactiva la escucha en un socket TCP.

working-directory (predeterminado: "/var/lib/redis")
Directorio en el que se almacenan los ficheros de base de datos y relacionados.

8.8.10 Servicios de correo

El módulo (gnu services mail) proporciona definiciones de servicios Guix para servicios de correo electrónico: servidores IMAP, POP3 y LMTP, así como agentes de transporte de correo (MTA). ¡Muchos acrónimos! Estos servicios se detallan en las subsecciones a continuación.
Servicio Dovecot

dovecot-service [#:config (dovecot-configuration)] [Procedimiento Scheme]

Devuelve un servicio que ejecuta el servidor de correo IMAP/POP3/LMTP Dovecot.

Habitualmente Dovecot no necesita mucha configuración; el objeto de configuración pre-determinado creado por (dovecot-configuration) es suficiente si su correo se entrega en ~/Maildir. Un certificado auto-firmado se generará para las conexiones protegidas por TLS, aunque Dovecot también escuchará en puertos sin cifrar de manera predeterminada. Existe un amplio número de opciones no obstante, las cuales las administradoras del correo pueden que deban cambiar, y como en el caso de otros servicios, Guix permite a la administradora del sistema la especificación de dichos parámetros a través de una interfaz Scheme uniforme.

Por ejemplo, para especificar que el correo se encuentra en maildir:~/.correo, se debe instanciar el servicio de Dovecot de esta manera:

(dovecot-service #:config (dovecot-configuration (mail-location "maildir:~/.correo")))

A continuación se encuentran los parámetros de configuración disponibles. El tipo de cada parámetro antecede la definición del mismo; por ejemplo, ’string-list foo’ indica que el parámetro foo debe especificarse como una lista de cadenas. También existe la posibilidad de especificar la configuración como una cadena, si tiene un fichero dovecot.conf antiguo que quiere trasladar a otro sistema; véase el final para más detalles.

Los campos disponibles de dovecot-configuration son:

package dovecot [parámetro de dovecot-configuration]

El paquete dovecot.

lista-cadenas-separada-comas listen [parámetro de dovecot-configuration]

Una lista de direcciones IP o nombres de máquina donde se escucharán conexiones. ‘*’ escucha en todas las interfaces IPv4, ‘::’ escucha en todas las interfaces IPv6. Si desea especificar puertos distintos a los predefinidos o algo más complejo, personalice los campos de dirección y puerto del ‘inet-listener’ de los servicios específicos en los que tenga interés.

lista-protocol-configuration [parámetro de dovecot-configuration]

protocols

Lista de protocolos que se desea ofrecer. Los protocolos disponibles incluyen ‘imap’, ‘pop3’ y ‘lmtp’.

Los campos disponibles de protocol-configuration son:

string name [parámetro de protocol-configuration]

El nombre del protocolo.

string auth-socket-path [parámetro de protocol-configuration]

Ruta del socket de UNIX del servidor de identificación maestro para la búsqueda de usuarias. Se usa por parte de imap (para usuarias compartidas) y lda. Su valor predeterminado es ‘"/var/run/dovecot/auth-userdb"’. 
Capítulo 8: Configuración del sistema

list-cadenas-separada-espacios\[parámetro de protocol-configuration\]
mail-plugins
Lista separada por espacios de módulos a cargar.

entero-no-negativo\[parámetro de protocol-configuration\]
mail-max-userip-connections
Número máximo de conexiones IMAP permitido para una usuaria desde cada dirección IP. ATENCIÓN: El nombre de usuaria es sensible a las mayúsculas. Su valor predeterminado es ‘10’.

lista-service-configuration\[parámetro de dovecot-configuration\]
services

Los campos disponibles de service-configuration son:

string kind\[parámetro de service-configuration\]
El tipo del servicio. Entre los valores aceptados se incluye director, imap-login, pop3-login, lmtp, imap, pop3, auth, auth-worker, dict, tcpwrap, quota-warning o cualquier otro.

lista-listener-configuration\[parámetro de service-configuration\]
listeners
Procesos de escucha para el servicio. Un proceso de escucha es un objeto unix-listener-configuration, un objeto fifo-listener-configuration o un objeto inet-listener-configuration. Su valor predeterminado es ‘()’.

Los campos disponibles de unix-listener-configuration son:

string path\[parámetro de unix-listener-configuration\]
Ruta al fichero, relativa al campo base-dir. También se usa como nombre de la sección.

string mode\[parámetro de unix-listener-configuration\]
Modo de acceso del socket. Su valor predeterminado es ‘”0600”’.

string user\[parámetro de unix-listener-configuration\]
Usuaria que posee el socket. Su valor predeterminado es ‘””’.

string group\[parámetro de unix-listener-configuration\]
Grupo que posee el socket. Su valor predeterminado es ‘””’.

Los campos disponibles de fifo-listener-configuration son:

string path\[parámetro de fifo-listener-configuration\]
Ruta al fichero, relativa al campo base-dir. También se usa como nombre de la sección.

string mode\[parámetro de fifo-listener-configuration\]
Modo de acceso del socket. Su valor predeterminado es ‘”0600”’.
string user [parámetro de fifo-listener-configuration]
   Usuaria que posee el socket. Su valor predeterminado es ‘“”’.

string group [parámetro de fifo-listener-configuration]
   Grupo que posee el socket. Su valor predeterminado es ‘“”’.

Los campos disponibles de inet-listener-configuration son:

string protocol [parámetro de inet-listener-configuration]
   El protocolo con el que se esperan las conexiones.

string address [parámetro de inet-listener-configuration]
   La dirección en la que se escuchará, o vacío para escuchar en todas las
direcciones. Su valor predeterminado es ‘“”’.

entero-no-negativo [parámetro de inet-listener-configuration]
   port
   El puerto en el que esperarán conexiones.

boolean ssl? [parámetro de inet-listener-configuration]
   Si se usará SSL para este servicio; ‘yes’ (sí), ‘no’ o ‘required’ (necesario).
   Su valor predeterminado es ‘#t’.

entero-no-negativo [parámetro de service-configuration]
   client-limit
   Número máximo de conexiones simultáneas por cliente por proceso. Una vez
se reciba este número de conexiones, la siguiente conexión entrante solicitará
da Dovecot el inicio de un nuevo proceso. Si se proporciona el valor 0, se usa
default-client-limit.
   El valor predeterminado es ‘0’.

entero-no-negativo [parámetro de service-configuration]
   service-count
   Número de conexiones a manejar antes de iniciar un nuevo proceso. Habitual-
mente los únicos valores útiles son 0 (ilimitadas) o 1. 1 es más seguro, pero 0

entero-no-negativo [parámetro de service-configuration]
   process-limit
   Número máximo de procesos que pueden existir para este servicio. Si se pro-
porciona el valor 0, se usa default-process-limit.
   El valor predeterminado es ‘0’.

entero-no-negativo [parámetro de service-configuration]
   process-min-avail
   Número de procesos que siempre permanecerán a la espera de más conexiones.
   Su valor predeterminado es ‘0’.

entero-no-negativo [parámetro de service-configuration]
   vsz-limit
   Si proporciona ‘service-count 0’, probablemente necesitará incrementar este
   valor. Su valor predeterminado es ‘256000000’.
Capítulo 8: Configuración del sistema

dict-configuration dict  [parámetro de dovecot-configuration]
Configuración de Dict, como la creada por el constructor dict-configuration.
Los campos disponibles de dict-configuration son:

campos-libres entries  [parámetro de dict-configuration]
Una lista de pares clave-valor que este diccionario debe incorporar. Su valor predeterminado es ‘(’.

lista-passdb-configuration passdbs  [parámetro de dovecot-configuration]
Una lista de configuraciones de passdb, cada una creada por el constructor passdb-configuration.
Los campos disponibles de passdb-configuration son:

string driver  [parámetro de passdb-configuration]

lista-cadenas-separada-espacios args  [parámetro de passdb-configuration]
Lista de parámetros separados por espacios para proporcionar al controlador passdb. Su valor predeterminado es ‘””’.

lista-userdb-configuration userdbs  [parámetro de dovecot-configuration]
Lista de configuraciones userdb, cada una creada por el constructor userdb-configuration.
Los campos disponibles de userdb-configuration son:

string driver  [parámetro de userdb-configuration]
El controlador que userdb debe usar. Entre los valores aceptados se incluye ‘passwd’ y ‘static’. Su valor predeterminado es ‘”passwd”’.

lista-cadenas-separada-espacios args  [parámetro de userdb-configuration]
Lista separada por espacios de parámetros usados para el controlador userdb. Su valor predeterminado es ‘””’.

parámetros-libres override-fields  [parámetro de userdb-configuration]
Sustituye los valores de campos de passwd. Su valor predeterminado es ‘(’.

plugin-configuration  [parámetro de dovecot-configuration]
plugin-configuration
Configuración del módulo, creada por el constructor plugin-configuration.

lista-namespace-configuration namespaces  [parámetro de dovecot-configuration]
Lista de espacios de nombres. Cada elemento de la lista debe crearse con el constructor namespace-configuration.
Los campos disponibles de namespace-configuration son:
string name [parámetro de namespace-configuration]
Nombre para este espacio de nombres.

string type [parámetro de namespace-configuration]
Tipo del espacio de nombres: ‘private’, ‘shared’ o ‘public’. Su valor predeterminado es ‘"private"’.

string separator [parámetro de namespace-configuration]
Separador jerárquico usado. Debe usar el mismo separador para todos los espacios de nombres o algunos clientes pueden confundirse. Habitualmente ‘/’ es un buen valor. El valor predeterminado depende no obstante del formato del almacenamiento de correo subyacente. Su valor predeterminado es ‘""’.

string prefix [parámetro de namespace-configuration]
Prefijo necesario para el acceso a este espacio de nombres. Tiene que ser necesario para todos los espacios de nombres. Por ejemplo ‘Public/’. Su valor predeterminado es ‘""’.

string location [parámetro de namespace-configuration]
Localización física de la bandeja de correo. En el mismo formato que la localización del correo, que también es su valor predeterminado. Su valor predeterminado es ‘""’.

boolean inbox? [parámetro de namespace-configuration]
Únicamente puede existir una bandeja de entrada (INBOX), y esta configuración define qué espacio de nombres la posee. Su valor predeterminado es ‘#f’.

boolean hidden? [parámetro de namespace-configuration]
Si un espacio de nombres está oculto, no se anuncia a los clientes a través de la extensión NAMESPACE. Lo más probable es que también desee proporcionar ‘list? #f’. Es útil principalmente durante la conversión desde otro servidor de correo con espacios de nombres distintos que desea marcar como obsoletos pero que todavía funcionan. Por ejemplo, puede crear espacios de nombres ocultos con prefijos ‘~/mail’, ‘~%u/mail’ y ‘mail/’. Su valor predeterminado es ‘#f’.

boolean list? [parámetro de namespace-configuration]
Muestra las bandejas de correo bajo este espacio de nombres con la orden LIST. Hace que el espacio de nombres sea visible para clientes que no permiten la extensión NAMESPACE. El valor especial children enumera las bandejas de correo descendientes, pero oculta el prefijo del espacio de nombres. Su valor predeterminado es ‘#t’.

boolean subscriptions? [parámetro de namespace-configuration]
El espacio de nombres maneja sus propias suscripciones. Si es #f, el espacio de nombres superior las maneja. El prefijo vacío siempre debe tener este valor como #t. Su valor predeterminado es ‘#t’.

lista-mailbox-configuration [parámetro de namespace-configuration]
mailboxes
Lista de bandejas de correo predefinidas en este espacio de nombres. Su valor predeterminado es ‘(’.

Los campos disponibles de `mailbox-configuration` son:

**string name** [parámetro de `mailbox-configuration`]
Nombre de esta bandeja de correo.

**string auto** [parámetro de `mailbox-configuration`]
Con ‘create’ se crea de forma automática esta bandeja de correo. Con ‘subscribe’ se crea y se suscribe a la bandeja de correo. Su valor predeterminado es ‘no’.

**lista-cadenas-separada-espacios** [parámetro de `mailbox-configuration`]
`special-use`
Lista de atributos SPECIAL-USE de IMAP como se especifican en el RFC 6154. Entre los valores aceptados se incluye `All`, `Archive`, `Drafts`, `Flagged`, `Junk`, `Sent` y `Trash`. Su valor predeterminado es ‘()’.

**nombre-fichero base-dir** [parámetro de `dovecot-configuration`]
Directorio base donde se almacenan los datos de tiempo de ejecución. Su valor predeterminado es ‘"/var/run/dovecot/"’.

**string login-greeting** [parámetro de `dovecot-configuration`]
Mensaje de saludo para clientes. Su valor predeterminado es ‘"Dovecot ready."’.

**lista-cadenas-separada-espacios** [parámetro de `dovecot-configuration`]
`login-trusted-networks`
Lista de rangos de red en los que se confía. Las conexiones desde estas IP tienen permitido forzar sus direcciones IP y puertos (para el registro y las comprobaciones de identidad). ‘disable-plaintext-auth’ también se ignora en estas redes. Habitualmente aquí se especificarían los servidores proxy IMAP. Su valor predeterminado es ‘()’.

**lista-cadenas-separada-espacios** [parámetro de `dovecot-configuration`]
`login-access-sockets`
Lista de sockets para la comprobación de acceso al sistema (por ejemplo tcpwrap). Su valor predeterminado es ‘()’.

**boolean verbose-proctitle?** [parámetro de `dovecot-configuration`]
Muestra títulos de procesamiento más detallados (en la postdata). Actualmente muestra el nombre de la usuaria y su dirección IP. Es útil para ver quién está usando procesos IMAP realmente (por ejemplo bandejas de correo compartidas o si el mismo identificador numérico de usuaria se usa para varias cuentas). Su valor predeterminado es ‘#f’.

**boolean shutdown-clients?** [parámetro de `dovecot-configuration`]
Determina si se deben finalizar todos los procesos cuando el proceso maestro de Dovecot termine su ejecución. El valor #f significa que Dovecot puede actualizarse sin forzar el cierre de las conexiones existentes (aunque esto puede ser un problema si la actualización se debe, por ejemplo, a la corrección de una vulnerabilidad). Su valor predeterminado es ‘#t’.
Capítulo 8: Configuración del sistema

**entero-no-negativo**

*doveadm-worker-count*

Si es no-cero, ejecuta las órdenes del correo a través de este número de conexiones al servidor doveadm, en vez de ejecutarlas directamente en el mismo proceso. Su valor predeterminado es ‘0’.

**string**

*doveadm-socket-path*

Socket UNIX o máquina:puerto usados para la conexión al servidor doveadm. Su valor predeterminado es ‘”doveadm-server”’.

**lista-cadenas-separadas-espacios**

*import-environment*

Lista de variables de entorno que se preservan al inicio de Dovecot y se proporcionan a los procesos iniciados. También puede proporcionar pares clave=valor para proporcionar siempre dicha configuración específica.

**boolean**

*disable-plaintext-auth?*

Desactiva la orden LOGIN y todas las otras identificaciones en texto plano a menos que se use SSL/TLS (capacidad LOGINDISABLED). Tenga en cuenta que si la IP remota coincide con la IP local (es decir, se ha conectado desde la misma máquina), la conexión se considera segura y se permite la identificación en texto plano. Véase también la configuración ssl=required. Su valor predeterminado es ‘#t’.

**entero-no-negativo**

*auth-cache-size*

Tamaño de la caché de identificaciones (por ejemplo, ‘#e10e6’. 0 significa que está desactivada. Tenga en cuenta que bsdauth, PAM y vpopmail necesitan un valor en ‘cache-key’ para que se use la caché. Su valor predeterminado es ‘0’.

**string**

*auth-cache-ttl*

Tiempo de vida de los datos almacenados en caché. Los registros de caché no se usan tras su expiración, *excepto* si la base de datos principal devuelve un fallo interno. También se intentan manejar los cambios de contraseña de manera automática: si la identificación previa de la usuaria fue satisfactoria, pero no esta última, la caché no se usa. En estos momentos únicamente funciona con identificación en texto plano. Su valor predeterminado es ‘”1 hour”’.

**string**

*auth-cache-negative-ttl*

Tiempo de vida para fallos de búsqueda en la caché (usuaria no encontrada, la contraseña no coincide). 0 desactiva completamente su almacenamiento en caché. Su valor predeterminado es ‘”1 hour”’.

**lista-cadenas-separadas-espacios**

*auth-realms*

Lista de dominios para los mecanismos de identificación de SASL que necesite. Puede dejarla vacía si no desea permitir múltiples dominios. Muchos clientes simplemente usarán el primero de la lista, por lo que debe mantener el dominio predeterminado en primera posición. Su valor predeterminado es ‘()’. 
string auth-default-realm [parámetro de dovecot-configuration]
Dominio predeterminado usado en caso de no especificar ninguno. Esto se usa tanto en dominios SASL como al añadir @dominio al nombre de usuaria en los ingresos al sistema a través de texto en claro. Su valor predeterminado es ‘”’.

string auth-username-chars [parámetro de dovecot-configuration]
Lista de caracteres permitidos en los nombres de usuaria. Si el nombre de usuaria proporcionado contiene un carácter no enumerado aquí, el login falla automáticamente. Es únicamente una comprobación adicional para asegurarse de que las usuarias no pueden explotar ninguna potencial vulnerabilidad con el escape de comillas en bases de datos SQL/LDAP. Si desea permitir todos los caracteres, establezca este valor a la cadena vacía. Su valor predeterminado es ‘"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890.-_@"’.

string auth-username-translation [parámetro de dovecot-configuration]
Traducciones de caracteres de nombres de usuaria antes de que se busque en las bases de datos. El valor contiene series de caracteres "original -> transformado". Por ejemplo ‘#@/@’ significa que ‘#’ y ‘/’ se traducen en ‘@’. Su valor predeterminado es ‘’.

string auth-username-format [parámetro de dovecot-configuration]
Formato proporcionado al nombre de usuaria antes de buscarlo en las bases de datos. Puede usar variables estándar aquí, por ejemplo %Lu transformará el nombre a minúsculas, %n eliminará el fragmento del dominio si se proporcionó, o ‘%n-AT-%d’ cambiaría ‘@’ en ‘-AT-’. Esta traducción se realiza tras los cambios de ‘auth-username-translation’. Su valor predeterminado es ‘%Lu’.

string auth-master-user-separator [parámetro de dovecot-configuration]
Si desea permitir que usuarias maestras ingresen mediante la especificación del nombre de usuaria maestra dentro de la cadena de nombre de usuaria normal (es decir, sin usar el mecanismo para ello implementado por SASL), puede especificar aquí el carácter separador. El formato es entonces <usuaria><separador>usuaria maestra>. UW-IMAP usa ‘*’ como separador, por lo que esa puede ser una buena elección. Su valor predeterminado es ‘”’.

string auth-anonymous-username [parámetro de dovecot-configuration]
Usuaria usada para las usuarias que ingresen al sistema con el mecanismo de SASL ANONYMOUS. Su valor predeterminado es ‘anonymous’.

entero-no-negativo [parámetro de dovecot-configuration]
auth-worker-max-count
Número máximo de procesos de trabajo dovecot-auth. Se usan para la ejecución de consultas bloqueantes a passdb y userdb (por ejemplo MySQL y PAM). Se crean y destruyen bajo demanda de manera automática. Su valor predeterminado es ‘30’.

string auth-gssapi-hostname [parámetro de dovecot-configuration]
Nombre de máquina usado en los nombres de GSSAPI principales. Por omisión se usa el nombre devuelto por gethostname(). Use ‘$ALL’ (con comillas) para permitir todas las entradas keytab. Su valor predeterminado es ‘”’.
Capítulo 8: Configuración del sistema

**string auth-krb5-keytab**  [parámetro de dovecot-configuration]

Keytab de Kerberos usado para el mecanismo GSSAPI. Si no se especifica ninguno, se usa el predeterminado del sistema (habitualmente /etc/krb5.keytab). Puede ser necesario cambiar el servicio auth para que se ejecute como root y tenga permisos de lectura sobre este fichero. Su valor predeterminado es ‘”’.

**boolean auth-use-winbind?**  [parámetro de dovecot-configuration]

Se identifica con NTLM y GSS-SPNEGO mediante el uso del daemon winbind de Samba y la herramienta auxiliar ‘ntlm-auth’. [doc/wiki/Authentication/Mechanisms/Winbind.txt](doc/wiki/Authentication/Mechanisms/Winbind.txt]. Su valor predeterminado es ‘#f’.

**nombre-fichero auth-winbind-helper-path**  [parámetro de dovecot-configuration]

Ruta al binario de la herramienta auxiliar ‘ntlm-auth’ de Samba. Su valor predeterminado es ‘”/usr/bin/ntlm_auth”’.

**string auth-failure-delay**  [parámetro de dovecot-configuration]

Tiempo de espera antes de responder a identificaciones fallidas. Su valor predeterminado es ‘”2 secs”’.

**boolean auth-ssl-require-client-cert?**  [parámetro de dovecot-configuration]

Es necesario un certificado de cliente SSL válido o falla la identificación. Su valor predeterminado es ‘#f’.

**boolean auth-ssl-username-from-cert?**  [parámetro de dovecot-configuration]

Toma el nombre de usuario del certificado de cliente SSL, mediante el uso de X509_NAME_get_text_by_NID(), que devuelve el nombre común (CommonName) del nombre de dominio (DN) del sujeto del certificado. Su valor predeterminado es ‘#f’.

**lista-cadenas-separada-espacios auth-mechanisms**  [parámetro de dovecot-configuration]


**lista-cadenas-separada-espacios director-servers**  [parámetro de dovecot-configuration]

Lista de IP o nombres de máquina de los servidores directores, incluyendo este mismo. Los puertos se pueden especificar como ip:puerto. El puerto predeterminado es el mismo que el usado por el servicio director ‘inet-listener’. Su valor predeterminado es ‘()’.

**lista-cadenas-separada-espacios director-mail-servers**  [parámetro de dovecot-configuration]

Lista de IP o nombres de máquina de los servidores motores de correo. Se permiten también rangos, como 10.0.0.10-10.0.0.30. Su valor predeterminado es ‘()’.
string director-user-expire  [parámetro de dovecot-configuration]
Por cuanto tiempo se redirige a las usuarias a un servidor específico tras pasar ese
tiempo sin conexiones. Su valor predeterminado es ‘"15 min"’.

string director-username-hash  [parámetro de dovecot-configuration]
Cómo se traduce el nombre de usuaria antes de aplicar el hash. Entre los valores útiles
se incluye %Lu si la usuaria puede ingresar en el sistema con o sin @dominio, %Ld si
las bandejas de correo se comparten dentro del dominio. Su valor predeterminado es
‘"%Lu"’.

string log-path  [parámetro de dovecot-configuration]
Fichero de registro usado para los mensajes de error. ’syslog’ los envía a syslog,
’"/dev(stderr)’ a la salida de error estándar. Su valor predeterminado es ‘"syslog"’.

string info-log-path  [parámetro de dovecot-configuration]
Fichero de registro usado para los mensajes informativos. Por omisión se usa
‘log-path’. Su valor predeterminado es ‘""’.

string debug-log-path  [parámetro de dovecot-configuration]
Fichero de registro usado para los mensajes de depuración. Por omisión se usa
‘info-log-path’. Su valor predeterminado es ‘""’.

string syslog-facility  [parámetro de dovecot-configuration]
Subsistema de syslog (facility) usado si se envía el registro a syslog. De manera habitu-
al, si desea que no se use ‘mail’, se usará local0..local7. Otros subsistemas estándar
también están implementados. Su valor predeterminado es ‘"mail"’.

boolean auth-verbose?  [parámetro de dovecot-configuration]
Registra los intentos de identificación infructuosos y las razones por los que fallaron.
Su valor predeterminado es ‘#f’.

string auth-verbose-passwords  [parámetro de dovecot-configuration]
En caso de no coincidir la contraseña, registra la contraseña que se intentó. Los va-
lores aceptados son ‘no”, “plain” y “sha1”. “sha1” puede ser útil para diferenciar
intentos de descubrir la contraseña por fuerza bruta frente a una usuaria simplemente
intentando la misma contraseña una y otra vez. También puede recortar el valor a n
caracteres mediante la adición de ‘":n” (por ejemplo “sha1:6”). Su valor predetermi-
nado es ‘"no"’.

boolean auth-debug?  [parámetro de dovecot-configuration]
Registros aún más detallados para facilitar la depuración. Muestra, por ejemplo, las
consultas SQL. Su valor predeterminado es ‘#f’.

boolean auth-debug-passwords?  [parámetro de dovecot-configuration]
En caso de no coincidir la contraseña, registra las contraseñas y esquema usadas de
manera que el problema se pueda depurar. La activación de este valor también provoca
la activación de ‘auth-debug’. Su valor predeterminado es ‘#f’.

boolean mail-debug?  [parámetro de dovecot-configuration]
Permite la depuración de los procesos de correo. Puede ayudarle a comprender los
motivos en caso de que Dovecot no encuentre sus correos. Su valor predeterminado
es ‘#f’.
boolean verbose-ssl? [parámetro de dovecot-configuration]
Muestra los errores a nivel de protocolo SSL. Su valor predeterminado es ‘#f’.

string log-timestamp [parámetro de dovecot-configuration]
Prefijo de cada línea registrada en el fichero. Los códigos % tienen el formato de strftime(3). Su valor predeterminado es ‘"\"%b %d %H:%M:%S ""’.

lista-cadenas-separada-espacios [parámetro de dovecot-configuration]
login-log-format-elements
Lista de elementos que se desea registrar. Los elementos que tengan valor de variable no-vacía se unen para la formación de una cadena separada por comas.

string login-log-format [parámetro de dovecot-configuration]
Formato del registro de ingresos al sistema. %s contiene la cadena ‘login-log-format-elements’, %$ contiene los datos que se desean registrar. Su valor predeterminado es ‘"%$: %s"’.

string mail-log-prefix [parámetro de dovecot-configuration]
Prefijo de los registros de procesos de correo. Véase doc/wiki/Variables.txt para obtener una lista de variables que puede usar. Su valor predeterminado es ‘"\"%s(%u)<%{pid}><%{session}>: ""’.

string deliver-log-format [parámetro de dovecot-configuration]
Formato usado para el registro de las entregas de correo. Puede usar las variables:

- %s Mensaje de estado de entrega (por ejemplo: ‘saved to INBOX’)
- %m Message-ID
- %s Asunto
- %f De (dirección)
- %p Tamaño físico
- %w Tamaño virtual.

El valor predeterminado es ‘"msgid=%m: %$"’.

string mail-location [parámetro de dovecot-configuration]
Localización de las bandejas de correo de las usuarias. El valor predeterminado está vacío, lo que significa que Dovecot intenta encontrar las bandejas de correo de manera automática. Esto no funciona si la usuaria no tiene todavía ningún correo, por lo que debe proporcionarle a Dovecot la ruta completa.

Si usa mbox, proporcionar una ruta al fichero de la bandeja entrada “INBOX” (por ejemplo /var/mail/%u) no es suficiente. También debe proporcionarle a Dovecot la ruta de otras bandejas de correo. Este es el llamado “directorio de correo raíz”, y debe ser la primera ruta proporcionada en la opción ‘mail-location’.

Existen algunas variables especiales que puede usar, por ejemplo:

- ‘%u’ nombre de usuaria
- ‘%n’ parte de la usuaria en usuaria@dominio, idéntica a %u si no existe dominio
Capítulo 8: Configuración del sistema

‘%d’ parte del dominio en usuaria@dominio, vacía si no existe dominio
‘%h’ directorio de la usuaria

Véase doc/wiki/Variables.txt para obtener una lista completa. Algunos ejemplos:

‘maildir:~/Correo’
‘mbox:~/correo:INBOX=/var/mail/%u’
‘mbox:/var/mail/%d/%1n/%n:INDEX=/var/indexes/%d/%1n/%’

El valor predeterminado es ‘”’.

string mail-uid [parámetro de dovecot-configuration]
Usuaria del sistema y grupo que realizan el acceso al correo. Si se usan varias, userdb puede forzar su valor al devolver campos uid o gid. Se pueden usar tanto identificadores numéricos como nominales. <doc/wiki/UserIds.txt>. Su valor predeterminado es ‘”’.

string mail-gid [parámetro de dovecot-configuration]
El valor predeterminado es ‘”’.

string mail-privileged-group [parámetro de dovecot-configuration]
Grupo para permitir de forma temporal operaciones privilegiadas. Actualmente esto se usa únicamente con la bandeja de entrada (INBOX), cuando su creación inicial o el bloqueo con fichero .lock falle. Habitualmente se le proporciona el valor "mail" para tener acceso a /var/mail. Su valor predeterminado es ‘”’.

string mail-access-groups [parámetro de dovecot-configuration]
Proporciona acceso a estos grupos suplementarios a los procesos de correo. Habitualmente se usan para configurar el acceso a bandejas de correo compartidas. Tenga en cuenta que puede ser peligroso establecerlos si las usuarias pueden crear enlaces simbólicos (por ejemplo si se configura el grupo "mail" aquí, ln -s /var/mail ~/mail/var puede permitir a una usuaria borrar las bandejas de correo de otras usuarias, o ln -s /bandeja/compartida/secret/a ~/mail/mibandeja permitiría su lectura). Su valor predeterminado es ‘”’.

boolean mail-full-filesystem-access? [parámetro de dovecot-configuration]
Permite a los clientes acceso completo al sistema de ficheros. No existen otros controles de acceso que los realizados por el sistema operativo para los identificadores UID/GID activos. Funciona tanto con maildir como con mbox, lo que permite la adición de un prefijo a los nombres de las bandejas de correo como /ruta/ o “usuaria/”. Su valor predeterminado es ‘#f’.

boolean mmap-disable? [parámetro de dovecot-configuration]
No usa mmap() en absoluto. Es necesario si almacena índices en sistemas de ficheros compartidos (NFS o sistemas de ficheros en cluster). Su valor predeterminado es ‘#f’.

boolean dotlock-use-excl? [parámetro de dovecot-configuration]
Confía en el correcto funcionamiento de ‘O_EXCL’ para la creación de ficheros de bloqueo dotlock. NFS implementa ‘O_EXCL’ desde la versión 3, por lo que debería ser seguro usarlo hoy en día de manera predeterminada. Su valor predeterminado es ‘#t’.
string mail-fsync  [parámetro de dovecot-configuration]
Cuando se usarán las llamadas fsync() o fdatasync():

  * optimized: Cuando sea necesario para evitar la pérdida de datos importantes
  * always: Útil con, por ejemplo, NFS, donde las escrituras con write() se retrasan
  * never: Nunca se usa (mejor rendimiento, pero los fallos pueden producir pérdida de datos)

Su valor predeterminado es ‘"optimized"’.

boolean mail-nfs-storage?  [parámetro de dovecot-configuration]
El almacenamiento del correo se encuentra en NFS. Proporcionar un valor afirmativo provoca que Dovecot vacíe la caché de NFS cuando sea necesario. Si únicamente está usando un servidor de correo único esto no es necesario. Su valor predeterminado es ‘#f’.

boolean mail-nfs-index?
Los ficheros de índice de correo también se encuentran en NFS. Para proporcionar un valor afirmativo es necesario también proporcionar ‘mmap-disable? #t’ y ‘fsync-disable? #f’. Su valor predeterminado es ‘#f’.

string lock-method  [parámetro de dovecot-configuration]
Método de bloqueo para los ficheros de índice. Las alternativas son fcntl, flock y dotlock. Dotlock utiliza técnicas que pueden provocar un consumo de E/S mayor que otros métodos de bloqueo. Usuarios de NFS: flock no funciona, recuerde que se debe cambiar ‘mmap-disable’. Su valor predeterminado es ‘"fcntl"’.

nombre-fichero mail-temp-dir  [parámetro de dovecot-configuration]
Directorio en el que LDA/LMTP almacena de manera temporal correos entrantes de de más de 128 kB. Su valor predeterminado es ‘"/tmp"’.

entero-no-negativo first-valid-uid  [parámetro de dovecot-configuration]
Rango de UID aceptado para las usuarias. Principalmente es para asegurarse de que las usuarias no pueden ingresar en el sistema como un daemon u otras usuarias del sistema. Tenga en cuenta que el binario de dovecot tiene código que impide el ingreso al sistema como root y no puede llevarse a cabo incluso aunque ‘first-valid-uid’ tenga el valor 0. Su valor predeterminado es ‘500’.

entero-no-negativo last-valid-uid  [parámetro de dovecot-configuration]
El valor predeterminado es ‘0’.

entero-no-negativo first-valid-gid  [parámetro de dovecot-configuration]
Rango de GID aceptado para las usuarias. Las usuarias que no posean GID válido como ID primario de grupo no tienen permitido el ingreso al sistema. Si la usuaria es miembro de grupos suplementarios con GID no válido, no se activan dichos grupos. Su valor predeterminado es ‘1’.

entero-no-negativo last-valid-gid  [parámetro de dovecot-configuration]
El valor predeterminado es ‘0’.
**entero-no-negativo**

**mail-max-keyword-length**

[parámetro de dovecot-configuration]

Longitud máxima permitida para nombres de palabras clave del correo. El límite actúa únicamente en la creación de nuevas palabras clave. Su valor predeterminado es ‘50’.

**lista-archivos-sep-dos-puntos**

**valid-chroot-dirs**

[parámetro de dovecot-configuration]

Lista de directorios bajo los que se permite realizar la llamada al sistema chroot a procesos de correo (es decir, /var/mail permitiría realizar la llamada también en /var/mail/sub/directorio). Esta configuración no afecta a la configuración de ‘login-chroot’, ‘mail-chroot’ o la de chroot para identificación. En caso de estar vacía, se ignora "/./" en los directorios de usuarias. AVISO: Nunca añada directorios que las usuarias locales puedan modificar, puesto que podría permitir vulnerar el control de acceso como root. Habitualmente esta opción se activa únicamente si no permite a las usuarias acceder a un shell. <doc/wiki/Chrooting.txt>. Su valor predeterminado es ‘(‘).

**string mail-chroot**

[parámetro de dovecot-configuration]

Directorio predeterminado de la llamada al sistema chroot para los procesos de correo. El valor puede forzarse para usuarias específicas en la base de datos proporcionando "/./" en el directorio de la usuaria (por ejemplo, /home/./usuaria llama a chroot con /home). Tenga en cuenta que habitualmente no es necesario llamar a chroot, Dovecot no permite a las usuarias el acceso a ficheros más allá de su directorio de correo en cualquier caso. Si sus directorios de usuaria contienen el directorio de chroot como prefijo, agregue "/./" al final de ‘mail-chroot’. <doc/wiki/Chrooting.txt>. Su valor predeterminado es ‘”’.

**nombre-fichero auth-socket-path**

[parámetro de dovecot-configuration]

Ruta al socket de UNIX al servidor maestro de identificación para la búsqueda de usuarias. Se usa por parte de imap (para usuarias compartidas) y lda. Su valor predeterminado es ‘"/var/run/dovecot/auth-userdb"’.

**nombre-fichero mail-plugin-dir**

[parámetro de dovecot-configuration]

Directorio en el que se buscarán módulos de correo. Su valor predeterminado es ‘"/usr/lib/dovecot"’.

**lista-cadena-separada-espacios**

**mail-plugins**

[parámetro de dovecot-configuration]

Lista de módulos cargados en todos los servicios. Los módulos específicos para IMAP, LDA, etc. se añaden en esta lista en sus propios ficheros .conf. Su valor predeterminado es ‘(‘).

**entero-no-negativo**

**mail-cache-min-mail-count**

[parámetro de dovecot-configuration]

El número mínimo de correos en una bandeja antes de que las actualizaciones se realicen en un fichero de caché. Permite optimizar el comportamiento de Dovecot para reducir la tasa de escritura en el disco, lo que produce un aumento de la tasa de lectura. Su valor predeterminado es ‘0’.
**string mailbox-idle-check-interval** [parámetro de dovecot-configuration]

Cuando se ejecute la orden IDLE, la bandeja de correo se comprueba de vez en cuando para comprobar si existen nuevos correos o se han producido otros cambios. Esta configuración define el tiempo mínimo entre dichas comprobaciones. Dovecot también puede usar dnottyf, inotify y kqueue para recibir información inmediata sobre los cambios que ocurren. Su valor predeterminado es ‘"30 secs"’.

**boolean mail-save-crlf?** [parámetro de dovecot-configuration]

Almacena los correos con CR+LF en vez de Únicamente LF. Provoca que el envío de dichos correos use menos los procesados, especialmente al usar la llamada al sistema sendfile() con Linux y FreeBSD. Pero también aumenta la tasa de E/S del disco, lo que puede ralentizar el proceso. También tenga en cuenta que si otro software lee las bandejas mbox/maildir, se pueden tratar de forma incorrecta los caracteres CR adicionales, lo que puede provocar problemas. Su valor predeterminado es ‘#f’.

**boolean maildir-stat-dirs?** [parámetro de dovecot-configuration]

De manera predeterminada la orden LIST devuelve todas las entradas en maildir cuyo nombre empiece en punto. La activación de esta opción hace que Dovecot únicamente devuelva las entradas que sean directorios. Esto se lleva a cabo llamando a stat() con cada entrada, lo que causa mayor E/S del disco. (En sistemas que proporcionen valor a ‘dirent->d_type’ esta comprobación no tiene coste alguno y se realiza siempre independientemente del valor configurado aquí). Su valor predeterminado es ‘#f’.

**boolean maildir-copy-with-hardlinks?** [parámetro de dovecot-configuration]

Cuando se copia un mensaje, se usan enlaces duros cuando sea posible. Esto mejora mucho el rendimiento, y es difícil que produzca algún efecto secundario. Su valor predeterminado es ‘#t’.

**boolean maildir-very-dirty-syncs?** [parámetro de dovecot-configuration]

Asume que Dovecot es el único MUA que accede a maildir: Recorre el directorio cur/ únicamente cuando cambie su mtime de manera inesperada o cuando no se pueda encontrar el correo de otra manera. Su valor predeterminado es ‘#f’.

**lista-cadena-separada-espacios mbox-read-locks** [parámetro de dovecot-configuration]

Qué métodos de bloqueo deben usarse para el bloque de mbox. Hay cuatro disponibles:

- **dotlock**   Crea un fichero <bandeja>.lock. Esta es la solución más antigua y más segura con NFS. Si desea usar un directorio como /var/mail/, las usuarias necesitarán acceso de escritura a dicho directorio.

- **dotlock-try**   Lo mismo que dotlock, pero si se produce un fallo debido a los permisos o a que no existe suficiente espacio en disco, simplemente se omite el bloqueo.

- **fcntl**   Use este a ser posible. Funciona también con NFS si se usa lockd.

- **flock**   Puede no existir en todos los sistemas. No funciona con NFS.
lockf Puede no existir en todos los sistemas. No funciona con NFS.
Puede usar múltiples métodos de bloqueo; en ese caso el orden en el que se declaran es importante para evitar situaciones de bloqueo mutuo en caso de que otros MTA/MUA usen también múltiples métodos de bloqueo. Algunos sistemas operativos no permiten el uso de varios de ellos de manera simultánea.

`lista-cadena-separada-espacios` [parámetro de dovecot-configuration]
`mbox-write-locks`

`string mbox-lock-timeout` [parámetro de dovecot-configuration]
Tiempo máximo esperado hasta el bloqueo (de todos los ficheros) antes de interrumpir la operación. Su valor predeterminado es "5 mins".

`string mbox-dotlock-change-timeout` [parámetro de dovecot-configuration]
Si existe el fichero dotlock pero la bandeja no se ha modificado de ninguna manera, ignora el fichero de bloqueo tras este tiempo. Su valor predeterminado es "2 mins".

`boolean mbox-dirty-syncs?` [parámetro de dovecot-configuration]
Cuando el fichero mbox cambie de manera inesperada, se tiene que leer completamente para encontrar los cambios. Si es grande puede tardar bastante tiempo. Devido a que el cambio habitualmente es un nuevo correo añadido al final, sería más rápido únicamente leer los correos nuevos. Si se activa esta opción, Dovecot hace esto pero de todos modos vuelve a leer el fichero mbox al completo cuando algo en la bandeja no se encuentra como se esperaba. La única desventaja real de esta configuración es que si otro MUA cambia las opciones de los mensajes, Dovecot no tiene constancia de ello de manera inmediata. Tenga en cuenta que una sincronización completa se lleva a cabo con las órdenes SELECT, EXAMINE, EXPUNGE y CHECK. Su valor predeterminado es '#t'.

`boolean mbox-very-dirty-syncs?` [parámetro de dovecot-configuration]
Como 'mbox-dirty-syncs', pero no realiza sincronizaciones completas tampoco con las órdenes SELECT, EXAMINE, EXPUNGE o CHECK. Si se configura este valor, 'mbox-dirty-syncs' se ignora. Su valor predeterminado es '#f'.

`boolean mbox-lazy-writes?` [parámetro de dovecot-configuration]
Retrasa la escritura de las cabeceras de mbox hasta que se realice una escritura completa sincronizada (con las órdenes EXPUNGE y CHECK, y al cerrar la bandeja de correo). Es útil especialmente con POP3, donde el cliente habitualmente borra todos los correos. La desventaja es que nuestros cambios no son visibles para otros MUA de manera inmediata. Su valor predeterminado es '#t'.

`entero-no-negativo mbox-min-index-size` [parámetro de dovecot-configuration]
Si el tamaño del fichero mbox es menor que este valor (por ejemplo, 100k), no escribe ficheros de índice. Si el fichero de índice ya existe, todavía se usa para la lectura, pero no se actualiza. Su valor predeterminado es '0'.

`entero-no-negativo mbox-rotate-size` [parámetro de dovecot-configuration]
Tamaño máximo del fichero dbox hasta su rotación. Su valor predeterminado es '10000000'.

Capítulo 8: Configuración del sistema

string mdbox-rotate-interval [parámetro de dovecot-configuration]
Antigüedad máxima del fichero dbox hasta que se produce su rotación. Habitualmente en días. El día comienza a medianoche, por lo que 1d = hoy, 2d = ayer, etcétera. 0 = comprobación desactivada. Su valor predeterminado es ‘"1d"’.

boolean mdbox-preallocate-space? [parámetro de dovecot-configuration]
Cuando se crean nuevos ficheros mdbox, reserva inmediatamente un tamaño de ‘mbox-rotate-size’. Actualmente esta configuración funciona únicamente en Linux con determinados sistemas de ficheros (ext4, xfs). Su valor predeterminado es ‘#f’.

string mail-attachment-dir [parámetro de dovecot-configuration]
sdbox y mdbox permiten el almacenamiento de adjuntos en ficheros externos, lo que permite también su almacenamiento único. Otros motores no lo permiten por el momento.
AVISOS: Esta característica todavía no se ha probado mucho. Su uso queda bajo su propia responsabilidad.
Directorio raíz donde almacenar los adjuntos de los correos. Se desactiva en caso de estar vacío. Su valor predeterminado es ‘""’.

entero-no-negativo mail-attachment-min-size [parámetro de dovecot-configuration]
Los adjuntos de menor tamaño que este valor no se almacenan externamente. También es posible la escritura de un módulo que deshabilite el almacenamiento externo de adjuntos específicos. Su valor predeterminado es ‘128000’.

string mail-attachment-fs [parámetro de dovecot-configuration]
Motor del sistema de ficheros usado para el almacenamiento de adjuntos:
posix Dovecot no lleva a cabo el SiS (aunque esto puede ayudar a la deduplicación del propio sistema de ficheros)
sis posix SiS con comparación inmediata byte-por-byte durante el almacenamiento.
sis-queue posix SiS mediante comparación retrasada y deduplicación.
El valor predeterminado es ‘"sis posix"’.

string mail-attachment-hash [parámetro de dovecot-configuration]
Formato del hash usado en los ficheros adjuntos. Puede añadir cualquier texto y variables: %{md4}, %{md5}, %{sha1}, %{sha256}, %{sha512}, %{size}. Las variables pueden reducirse, por ejemplo %{sha256:80} devuelve únicamente los primeros 80 bits. Su valor predeterminado es ‘"%{sha1}"’.

entero-no-negativo default-process-limit [parámetro de dovecot-configuration]
El valor predeterminado es ‘100’.

entero-no-negativo default-client-limit [parámetro de dovecot-configuration]
El valor predeterminado es ‘1000’.
Capítulo 8: Configuración del sistema

**entero-no-negativo**  
[parámetro de `dovecot-configuration`]

**default-vsz-limit**
Límite predeterminado del tamaño de memoria virtual (VSZ) para procesos del servicio. Esta principalmente orientado a la captura y parada de procesos que pierden memoria antes de que utilicen toda la disponible. Su valor predeterminado es ‘256000000’.

**string default-login-user**  
[parámetro de `dovecot-configuration`]

Usuario de ingreso al sistema para los procesos de ingreso al sistema. Es la usuario en la que menos se confía en el sistema Dovecot. No debería tener acceso a nada en absoluto. Su valor predeterminado es ‘“dovenull”’.

**string default-internal-user**  
[parámetro de `dovecot-configuration`]

Usuario interna usada por procesos sin privilegios. Debería ser distinta a la usuario de ingreso, de modo que los procesos de ingreso al sistema no interfieran con otros procesos. Su valor predeterminado es ‘“dovecot”’.

**string ssl?**  
[parámetro de `dovecot-configuration`]


**string ssl-cert**  
[parámetro de `dovecot-configuration`]

Certificado X.509 de SSL/TLS codificado con PEM (clave pública). Su valor predeterminado es ‘”/etc/dovecot/default.pem”’.

**string ssl-key**  
[parámetro de `dovecot-configuration`]

Clave privada de SSL/TLS codificada con PEM. La clave se abre antes de renunciar a los privilegios de root, por lo que debe mantenerse legible únicamente para root. Su valor predeterminado es ‘”/etc/dovecot/private/default.pem”’.

**string ssl-key-password**  
[parámetro de `dovecot-configuration`]

Si el fichero de la clave está protegido por contraseña, introduzca dicha contraseña aquí. De manera alternativa, puede proporcionarla al iniciar dovecot con el parámetro -p. Como este fichero es habitualmente legible por todo el mundo, puede que desee desplazar esta opción a un fichero diferente. Su valor predeterminado es ‘””’.

**string ssl-ca**  
[parámetro de `dovecot-configuration`]

Certificado usado como autoridad de certificación de confianza codificado en PEM. Configure este valor únicamente si tiene intención de usar ‘ssl-verify-client-cert? #t’. El fichero debe contener el fichero de la o las AC seguido de las CRL correspondientes (por ejemplo, ‘ssl-ca </etc/ssl/certs/ca.pem’). Su valor predeterminado es ‘””’.

**boolean ssl-require-crl?**  
[parámetro de `dovecot-configuration`]

Es necesario que la comprobación de CRL sea satisfactoria para certificados de clientes.. Su valor predeterminado es ‘#t’.

**boolean ssl-verify-client-cert?**  
[parámetro de `dovecot-configuration`]

Solicita al cliente el envío de un certificado. Si también desea que sea un requisito, proporcione ‘auth-ssl-require-client-cert? #t’ en la sección de identificación. Su valor predeterminado es ‘#f’.
string ssl-cert-username-field  [parámetro de dovecot-configuration]
  Cual es el campo del certificado que determina el nombre de usuario. “commonName”
y “x500UniqueIdentifier” son las opciones habituales. También tendrá proporcionar
‘auth-ssl-username-from-cert? #t’. Su valor predeterminado es ‘"commonName"’.

string ssl-min-protocol  [parámetro de dovecot-configuration]
  Versión mínima aceptada del protocolo SSL. Su valor predeterminado es ‘"TLSv1"’.

string ssl-cipher-list  [parámetro de dovecot-configuration]
  Protocolos de cifrado de SSL usados. Su valor predeterminado es
  ‘ALL:!kRSA:!SRP:!kDHd:!DSS:!aNULL:!eNULL:!EXPORT:!DES:!3DES:!MD5:!PSK:!RC4:!ADH:!LOW@STRENGTH’.

string ssl-crypto-device  [parámetro de dovecot-configuration]
  Dispositivo de cifrado de SSL usado, ejecute “openssl engine” para obtener los valores
  aceptados. Su valor predeterminado es ‘""’.

string postmaster-address  [parámetro de dovecot-configuration]
  Dirección usada cuando se notifiquen correos rechazados. %d expande al dominio
  receptor. Su valor predeterminado es ‘"postmaster@%d"’.

string hostname  [parámetro de dovecot-configuration]
  Nombre de máquina usado en diversas partes de los correos enviados (por ejemplo, en
  Message-Id) y en las respuestas LMTP. Su valor predeterminado es ‘<el nombre real
  de la máquina>@dominio’. Su valor predeterminado es ‘""’.

boolean quota-full-tempfail?  [parámetro de dovecot-configuration]
  Si la usuaria supera la cuota, devuelve un fallo temporal en vez de rechazar el correo.
  Su valor predeterminado es ‘#f’.

nombre-fichero sendmail-path  [parámetro de dovecot-configuration]
  Binario usado para el envío de correos. Su valor predeterminado es
  ‘"/usr/sbin/sendmail"’.

string submission-host  [parámetro de dovecot-configuration]
  Si no está vacío, envía el correo a través de esta máquina[:puerto] SMTP en vez de
  usar sendmail Su valor predeterminado es ‘""’.

string rejection-subject  [parámetro de dovecot-configuration]
  Asunto: cabecera usada en el rechazo de correos. Puede usar las mismas variables
  que las indicadas en ‘rejection-reason’ a continuación. Su valor predeterminado es
  ‘"Rejected: %s"’.

string rejection-reason  [parámetro de dovecot-configuration]
  Mensaje de error legible por personas para el rechazo de correos. Puede usar variables:

  %n  CRLF
  %r  razón
  %s  asunto original
  %t  receptora

  El valor predeterminado es ‘"Your message to <%t> was automatically
  rejected:%n%r"’.
Capítulo 8: Configuración del sistema

string recipient-delimiter [parámetro de dovecot-configuration]
Carácter delimitador entre la parte local y el detalle en las direcciones de correo. Su valor predeterminado es ‘"+"’.

string lda-original-recipient-header [parámetro de dovecot-configuration]
Cabecera de donde se obtiene la dirección receptora original (la dirección de SMTP RCPT TO:) en caso de no estar disponible en otro lugar. El parámetro -a de dovecot-lda reemplaza este valor. Una cabecera usada para esto de manera común es X-Original-To. Su valor predeterminado es ‘""’.

boolean lda-mailbox-autocreate? [parámetro de dovecot-configuration]
¿Se debe crear una bandeja de correo no existente de manera automática al almacenar un correo? Su valor predeterminado es ‘#f’.

boolean lda-mailbox-autosubscribe? [parámetro de dovecot-configuration]
¿También se deben crear suscripciones de manera automática a las bandejas de correo creadas? Su valor predeterminado es ‘#f’.

entero-no-negativo imap-max-line-length [parámetro de dovecot-configuration]
Longitud máxima de la línea de órdenes de IMAP. Algunos clientes generan líneas de órdenes muy largas con bandejas de correo normales, por lo que debe incrementarlo si recibe los errores “Too long argument” (parámetro demasiado largo) o "IMAP command line too large" (línea de órdenes de IMAP demasiado grande). Su valor predeterminado es ‘64000’.

string imap-logout-format [parámetro de dovecot-configuration]
Formato de la cadena de IMAP de salida del sistema:

%i número total de bytes leídos del cliente
%o número total de bytes enviados al cliente.

Véase doc/wiki/Variables.txt para obtener una lista completa de todas las variables que puede usar. Su valor predeterminado es ‘"in=%i out=%o deleted=%{deleted} expunged=%{expunged} trashed=%{trashed} hdr_count=%{fetch_hdr_count} hdr_bytes=%{fetch_hdr_bytes} body_count=%{fetch_body_count} body_bytes=%{fetch_body_bytes}"’.

string imap-capability [parámetro de dovecot-configuration]
Fuerza el valor de la respuesta de IMAP CAPABILITY. Si el valor comienza con ‘+’, añade las capacidades especificadas sobre las predeterminadas (por ejemplo, +XFOO XBAR). Su valor predeterminado es ‘""’.

string imap-idle-notify-interval [parámetro de dovecot-configuration]
Durante cuanto tiempo se espera entre notificaciones "OK Still here" cuando el cliente se encuentre en estado IDLE. Su valor predeterminado es ‘"2 mins"’.

string imap-id-send [parámetro de dovecot-configuration]
Nombres y valores de campos de identificación (ID) que se enviarán a los clientes. El uso de * como un valor hace que Dovecot utilice el valor predeterminado. Los
Capítulo 8: Configuración del sistema 270

siguientes campos tienen actualmente valores predeterminados: name, version, os, os-version, support-url, support-email. Su valor predeterminado es "".

**string imap-id-log** [parámetro de dovecot-configuration]
Campos de identificación (ID) enviados para su registro por cliente. * significa todos. Su valor predeterminado es "".

**lista-cadena-separada-espacios** [parámetro de dovecot-configuration]
imap-client-workarounds
Soluciones temporales para varios errores de clientes:

delay-newmail
Envía notificaciones de nuevo correo EXISTS/RECENT únicamente en respuesta a ordenes NOOP o CHECK. Algunos clientes las ignoran en otro caso, por ejemplo OS Mail (<v2.1). Outlook Express tiene problemas mayores no obstante, sin esto puede mostrar a la usuaria errores "Message no longer in server". Tenga en cuenta que OE6 también falla con esta solución temporal si la sincronización se establece como "Headers Only".

**tb-extra-mailbox-sep**
Thunderbird se confunde de algún modo con LAYOUT=fs (mbox y dbox) y añade sufijos ‘/’ adicionales a los nombres de las bandejas de correo. Esta opción hace que Dovecot ignore el carácter ‘/’ adicional en vez de tratarlo como un nombre de bandeja de correo no válido.

**tb-lsub-flags**
Muestra las opciones \Noselect para respuestas LSUB con LAYOUT=fs (por ejemplo mbox). Esto permite a Thunderbird ser consciente de que no se pueden seleccionar y mostrarlas en gris, en vez de únicamente mostrar después el mensaje de error "no seleccionable".

El valor predeterminado es ‘()’.

**string imap-urlauth-host** [parámetro de dovecot-configuration]
Máquina permitida en las URL URLAUTH enviadas por el cliente. "*" permite todas. Su valor predeterminado es "".

¡Miau! Muchas opciones de configuración. Lo bueno es que Guix tiene una interfaz completa al lenguaje de configuración de Dovecot. Esto no permite únicamente declarar configuraciones de forma bonita, sino que también ofrece capacidades reflexivas: las usuarias pueden escribir código en Scheme para inspeccionar y transformar configuraciones.

No obstante, puede ser que únicamente desee usar un fichero dovecot.conf existente. En ese caso, puede proporcionar un objeto opaque-dovecot-configuration como parámetro #:config a dovecot-service. Como su nombre en inglés indica, una configuración opaca no tiene gran capacidad reflexiva.

Los campos disponibles de opaque-dovecot-configuration son:

**package dovecot** [parámetro de opaque-dovecot-configuration]
El paquete dovecot.
string string [parámetro de opaque-dovecot-configuration]
El contenido de dovecot.conf, como una cadena.

Por ejemplo, si su dovecot.conf fuese simplemente la cadena vacía, podría instanciar un servicio dovecot de esta manera:
(dovecot-service #:config
 (opaque-dovecot-configuration
  (string "")))

Servicio OpenSMTPD

opensmtpd-service-type [Variable Scheme]
Es el tipo del servicio OpenSMTPD (https://www.opensmtpd.org), cuyo valor debe ser un objeto opensmtpd-configuration como en este ejemplo:
(service opensmtpd-service-type
 (opensmtpd-configuration
  (config-file (local-file "/.mi-smtpd.conf"))))

opensmtpd-configuration [Tipo de datos]
Tipo de datos que representa la configuración de opensmtpd.

package (predeterminado: opensmtpd)
El objeto paquete del servidor SMTP OpenSMTPD.

config-file (predeterminado: %default-opensmtpd-file)
Objeto “tipo-fichero” del fichero de configuración de OpenSMTPD usado. De manera predeterminada escucha en la interfaz de red local, y pone a disposición de usuarias y daemon de la máquina local el servicio de correo, así como el envío de correo a servidores remotos. Ejecute man smtpd.conf para obtener más información.

Servicio Exim

exim-service-type [Variable Scheme]
Este es el tipo del agente de transferencia de correo (MTA) Exim (https://exim.org), cuyo valor debe ser un objeto exim-configuration como en este ejemplo:
(service exim-service-type
 (exim-configuration
  (config-file (local-file "/.mi-exim.conf"))))

Para usar un servicio exim-service-type debe tener también un servicio mail-aliases-service-type presente en su declaración operating-system (incluso aunque no exista ningún alias).

exim-configuration [Tipo de datos]
Tipo de datos que representa la configuración de exim.

package (predeterminado: exim)
El objeto paquete del servidor Exim.
Capítulo 8: Configuración del sistema

config-file (predeterminado: #f)
Objeto “tipo-fichero” del fichero de configuración de Exim usado. Si su valor es #f se usa el fichero de configuración predefinido proporcionado por el paquete en package. El fichero de configuración resultante se carga tras fijar las variables de configuración exim_user y exim_group.

Servicio Getmail

getmail-service-type [Variable Scheme]
El tipo del receptor de correo Getmail (http://pyropus.ca/software/getmail/), cuyo valor debe ser un objeto getmail-configuration.

Los campos disponibles de getmail-configuration son:

símbolo name [parámetro de getmail-configuration]
Un símbolo que identifique el servicio getmail.
El valor predeterminado es '"unset"'.

package package [parámetro de getmail-configuration]
El paquete getmail usado.

string user [parámetro de getmail-configuration]
Usuaria que ejecuta getmail.
El valor predeterminado es '"getmail"'.

string group [parámetro de getmail-configuration]
Grupo que ejecuta getmail.
El valor predeterminado es '"getmail"'.

string directory [parámetro de getmail-configuration]
El directorio usado para getmail.
El valor predeterminado es '"/var/lib/getmail/default"'.

getmail-configuration-file rcfile [parámetro de getmail-configuration]
El fichero de configuración de getmail usado.

Los campos disponibles de getmail-configuration-file son:

getmail-retriever-configuration [parámetro de getmail-configuration-file]
retriever
De qué cuenta de correo obtener el correo, y cómo acceder a dicha cuenta.

Los campos disponibles de getmail-retriever-configuration son:

string type [parámetro de getmail-retriever-configuration]
El controlador que userdb debe usar. Entre los valores aceptados se incluye ‘passwd’ y ‘static’.
El valor predeterminado es '"SimpleIMAPSSLRetriever"'.

string server [parámetro de getmail-retriever-configuration]
Usuaria que ejecutará el servidor de correo.
El valor predeterminado es ‘unset’.
string username [parámetro de getmail-retriever-configuration]
Usaria que ejecutará el servidor de correo.
El valor predeterminado es ‘unset’.

entero-no-negativo [parámetro de getmail-retriever-configuration]
port
Número de puerto al que conectarse.
El valor predeterminado es ‘#f’

string password [parámetro de getmail-retriever-configuration]
Sustituye los valores de campos de passwd.
El valor predeterminado es ‘”’.

list [parámetro de getmail-retriever-configuration]
password-command
Sustituye los valores de campos de passwd.
El valor predeterminado es ‘()’.

string keyfile [parámetro de getmail-retriever-configuration]
Fichero de claves con formato PEM usado para la negociación TLS.
El valor predeterminado es ‘”’.

string certfile [parámetro de getmail-retriever-configuration]
Fichero de certificado con formato PEM usado para la negociación TLS.
El valor predeterminado es ‘”’.

string ca-certs [parámetro de getmail-retriever-configuration]
Certificados de autoridad de certificación (CA) usados.
El valor predeterminado es ‘”’.

parameter-alist [parámetro de getmail-retriever-configuration]
extra-parameters
Parámetros adicionales del receptor de correo.
El valor predeterminado es ‘()’.

getmail-destination-configuration [parámetro de getmail-configuration-file]

destination
Qué hacer con los mensajes obtenidos.
Los campos disponibles de getmail-destination-configuration son:

string type [parámetro de getmail-destination-configuration]
Tipo de destino del correo. Entre los valores válidos se incluye ‘Maildir’, ‘Mboxrd’ y ‘MDA_external’.
El valor predeterminado es ‘unset’.

string-or-filelike [parámetro de getmail-destination-configuration]
path
Opción de ruta para el destino del correo. El comportamiento depende del tipo seleccionado.
El valor predeterminado es ‘”’.
**parámetro-alist** [parámetro de getmail-destination-configuration]

**extra-parameters**
Parámetros adicionales del destino.
El valor predeterminado es ‘()’.

**getmail-options-configuration** [parámetro de getmail-configuration-file]

**options**
Configuración de getmail.
Los campos disponibles de **getmail-options-configuration** son:

**entero-no-negativo** [parámetro de getmail-options-configuration]

**verbose**
Si tiene valor ‘0’, getmail únicamente imprimirá mensajes de aviso y errores. El valor ‘1’ significa que se imprimirán mensajes acerca de la recepción y el borrado de mensajes. Si tiene valor ‘2’, getmail imprimirá mensajes con cada una de sus acciones.
El valor predeterminado es ‘1’.

**boolean read-all** [parámetro de getmail-options-configuration]
Si es verdadero, getmail obtendrá todos los mensajes disponibles. En otro caso, únicamente recupera mensajes que no se hayan visto previamente.
El valor predeterminado es ‘#t’

**boolean delete** [parámetro de getmail-options-configuration]
Si se proporciona un valor verdadero, los mensajes se borrarán del servidor tras su recuperación y entrega posterior satisfactoria. En otro caso, los mensajes permanecerán en el servidor.
El valor predeterminado es ‘#f’

**entero-no-negativo** [parámetro de getmail-options-configuration]

**delete-after**
Getmail borrará los mensajes tras este número de días después de haberlos visto, si han sido entregados. Esto significa que los mensajes se mantendrán en el servidor este número de días tras entregarlos. El valor ‘0’ desactiva esta característica.
El valor predeterminado es ‘0’.

**entero-no-negativo** [parámetro de getmail-options-configuration]

**delete-bigger-than**
Borra los mensajes de tamaño mayor que estos bytes tras recibirlos, incluso si las opciones “delete” y “delete-after” están desactivadas. El valor ‘0’ desactiva esta característica.
El valor predeterminado es ‘0’.

**entero-no-negativo** [parámetro de getmail-options-configuration]

**max-bytes-per-session**
Obtiene mensajes hasta este número de bytes en total antes de cerrar la sesión con el servidor. El valor ‘0’ desactiva esta característica.
El valor predeterminado es ‘0’.
Capítulo 8: Configuración del sistema

**entero-no-negativo** [parámetro de getmail-options-configuration]

**max-message-size**

No obtiene mensajes con mayor tamaño que este número de bytes. El valor ‘0’ desactiva esta característica.

El valor predeterminado es ‘0’.

**boolean** [parámetro de getmail-options-configuration]

**delivered-to**

Si es verdadero, getmail añadirá una cabecera “Delivered-To” a los mensajes.

El valor predeterminado es ‘#t’

**boolean** received [parámetro de getmail-options-configuration]

Si es verdadero, getmail añadirá una cabecera “Received” a los mensajes.

El valor predeterminado es ‘#t’

**string** message-log [parámetro de getmail-options-configuration]

Getmail generará un registro de sus acciones en el fichero nombrado. El valor ‘””’ desactiva esta característica.

El valor predeterminado es ‘””’

**boolean** message-log-syslog [parámetro de getmail-options-configuration]

Si es verdadero, getmail registrará sus acciones a través del registro del sistema.

El valor predeterminado es ‘#f’

**boolean** message-log-verbose [parámetro de getmail-options-configuration]

Si es verdadero, getmail registraría información sobre los mensajes que no se hayan podido recuperar y la razón para no hacerlo, así como líneas de inicio y fin informativas.

El valor predeterminado es ‘#f’

**parameter-alist** extra-parameters [parámetro de getmail-options-configuration]

Opciones adicionales a incluir.

El valor predeterminado es ‘()’.

**lista idle** [parámetro de getmail-configuration]

Una lista de bandejas de correo en las que getmail debe esperar en el servidor nuevas notificaciones de correo. Esto depende de que el servidor implemente la extensión IDLE.

El valor predeterminado es ‘()’.

**lista environment-variables** [parámetro de getmail-configuration]

Variables de entorno proporcionadas a getmail.

El valor predeterminado es ‘()’.
Servicios de alias de correo

mail-aliases-service-type  
Este es el tipo del servicio que proporciona /etc/aliases, donde se especifica cómo entregar el correo a las usuarias de este sistema.

(service mail-aliases-service-type  
'(("postmaster" "rober")  
("rober" "rober@example.com" "rober@example2.com")))

La configuración del servicio mail-aliases-service-type es una lista asociada que indica cómo se debe entregar el correo que viene del sistema. Cada entrada tiene la forma (alias direcciones ...), siendo alias un alias (nombre) local y direcciones específica dónde se debe entregar el correo de esta usuaria.

No es necesario que los alias existan como usuarias en el sistema local. En el ejemplo previo, no es necesario que exista una entrada postmaster en el campo user-accounts de operating-system para que el correo de postmaster se entregue a rober (que a su vez entregará el correo a rober@example.com y rober@example2.com).

daemon de IMAP3 de GNU Mailutils

imap4d-service-type  
Es el tipo del daemon IMAP4 de GNU Mailutils (véase Sección “imap4d” en GNU Mailutils Manual), cuyo valor debe ser un objeto imap4d-configuration como en este ejemplo:

(service imap4d-service-type  
(imap4d-configuration  
(config-file (local-file "imap4d.conf"))))

imap4d-configuration  
Tipo de datos que representa la configuración de imap4d.

package (predeterminado: mailutils)  
El paquete que proporciona imap4d.

cfg-file (predeterminado: %default-imap4d-config-file)  
Objeto “tipo-fichero” con el fichero de configuración usado, de manera predeterminada escucha en el puerto TCP 143 de localhost. Véase Sección “Conf-imap4d” en GNU Mailutils Manual, para más detalles.

8.8.11 Servicios de mensajería

El módulo (gnu services messaging) proporciona definiciones de servicios Guix para servicios de mensajería: actualmente únicamente Prosody está implementado.

Servicio Prosody

prosody-service-type  
Este es el tipo para el servidor de comunicaciones XMPP Prosody (https://prosody.im). Su valor debe ser un registro prosody-configuration como en este ejemplo:

(service prosody-service-type
(prosody-configuration
   (modules-enabled (cons* "groups" "mam" %default-modules-enabled))
   (int-components
      (list
         (int-component-configuration
            (hostname "conference.example.net")
            (plugin "muc")
            (mod-muc (mod-muc-configuration)))))
   (virtualhosts
      (list
         (virtualhost-configuration
            (domain "example.net")))))

Véase a continuación detalles acerca de *prosody-configuration*.

De manera predeterminada, Prosody no necesita demasiada configuración. Únicamente un campo *virtualhost* es necesario: especifica el dominio en el que se desea que Prosody proporcione el servicio.

Puede realizar varias comprobaciones preliminares sobre la configuración generada con la orden *prosodyctl check*.

Prosodyctl también le ayudará con la importación de certificados del directorio *letsencrypt* de modo que la usuaria *prosody* pueda acceder a ellos. Véase https://prosody.im/doc/letsencrypt.

    prosodyctl --root cert import /etc/letsencrypt/live

A continuación se encuentran los parámetros de configuración disponibles. El tipo de cada parámetro antecede la definición del mismo; por ejemplo, *string-list foo* indica que el parámetro *foo* debe especificarse como una lista de cadenas. Los tipos que empiezan con *maybe-* identifican parámetros que no aparecerán en *prosody.cfg.lua* cuando su valor sea *disabled*.

También existe la posibilidad de especificar la configuración como una cadena, por si tiene un fichero *prosody.cfg.lua* antiguo que desea transportar desde otro sistema; véase más detalles al final.

El tipo *file-object* designa o bien un objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108) o un nombre de fichero.

Los campos disponibles de *prosody-configuration* son:

**package prosody** [parámetro de *prosody-configuration*]
El paquete Prosody.

**nombre-fichero data-path** [parámetro de *prosody-configuration*]
Ruta del directorio de almacenamiento de datos de Prosody. Véase https://prosody.im/doc/configure. Su valor predeterminado es ‘"/var/lib/prosody"’.

**lista-file-object plugin-paths** [parámetro de *prosody-configuration*]
nombre-fichero certificates [parámetro de prosody-configuration]
Cada máquina virtual y componente necesitan un certificado de manera que los clientes y servidores puedan verificar su identidad de manera segura. Prosody cargará de manera automática certificados/claves del directorio especificado aquí. Su valor predeterminado es "/etc/prosody/certs".

lista-string admins [parámetro de prosody-configuration]
Es una lista de cuentas con permisos de administración en el servidor. Tenga en cuenta que debe crear las cuentas de manera separada. Véase https://prosody.im/doc/admins y https://prosody.im/doc/creating_accounts. Ejemplo: (admins '"usuaria1@example.com" "usuaria2@example.net") Su valor predeterminado es ‘()’.

boolean use-libevent? [parámetro de prosody-configuration]

lista-módulos modules-enabled [parámetro de prosody-configuration]
La lista de módulos de Prosody cargada durante el arranque. Busca en el fichero de módulos mod_nombremodulo.lua, por lo que asegúrese de que también exista. La documentación de los módulos puede encontrarse en: https://prosody.im/doc/modules. Su valor predeterminado es ‘("roster" "saslauth" "tls" "dialback" "disco" "carbons" "private" "blocklist" "vcard" "version" "uptime" "time" "ping" "pep" "register" "admin_adhoc")’.

lista-string modules-disabled [parámetro de prosody-configuration]
‘"offline", "c2s" y "s2s" se cargan de manera automática, pero puede desactivarlos si los añade a esta lista. Su valor predeterminado es ‘()’.

file-object groups-file [parámetro de prosody-configuration]
Ruta a un fichero de texto donde se definan los grupos compartidos. Si esta ruta está vacía, ‘mod_groups’ no hace nada. Véase https://prosody.im/doc/modules/mod_groups. Su valor predeterminado es ‘”/var/lib/prosody/sharedgroups.txt”’.

boolean allow-registration? [parámetro de prosody-configuration]

maybe-ssl-configuration ssl [parámetro de prosody-configuration]
Estas opciones de configuración están relacionadas con SSL/TLS. La mayor parte no se proporcionan para usar los valores predeterminados de Prosody. Si no entiende completamente estas opciones, no las añada a su configuración, es fácil aumentar la vulnerabilidad de su servidor si las usa. Véase https://prosody.im/doc/advanced_ssl_config.
Los campos disponibles de ssl-configuration son:

maybe-string protocol [parámetro de ssl-configuration]
Determina el inicio del protocolo usado (handshake).
maybe-nombre-fichero key [parámetro de ssl-configuration]
Ruta a su fichero de clave privada.

maybe-nombre-fichero certificate [parámetro de ssl-configuration]
Ruta al fichero de su certificado.

file-object capath [parámetro de ssl-configuration]
Ruta al directorio que contiene los certificados raíz en los que desea que Prosody confíe al verificar los certificados de servidores remotos. Su valor predeterminado es "'/etc/ssl/certs".

maybe-file-object cafile [parámetro de ssl-configuration]
Ruta al fichero que contiene los certificados raíz en los que desea que Prosody confíe. Es similar a capath pero con todos los certificados concatenados en el mismo fichero.

maybe-lista-string verify [parámetro de ssl-configuration]
Una lista de opciones de verificación (son de manera prácticamente directa las opciones de set_verify() de OpenSSL).

maybe-lista-string options [parámetro de ssl-configuration]
Una lista de opciones generales relacionadas con SSL/TLS. Se relacionan directamente con las opciones set_options() de OpenSSL. Para obtener una lista completa de las opciones disponibles en LuaSec, véase los ficheros de fuentes de LuaSec.

maybe-entero-no-negativo depth [parámetro de ssl-configuration]
Cómo de larga puede ser la cadena de autoridades de certificación a comprobar cuando se busque un certificado de confianza.

maybe-string ciphers [parámetro de ssl-configuration]
Una cadena de algoritmos de cifrado de OpenSSL. Selecciona que algoritmos ofrecerá Prosody a los clientes, y en qué orden.

maybe-nombre-fichero dhparam [parámetro de ssl-configuration]
Una ruta a un fichero que contenga parámetros para el intercambio de claves Diffie-Hellman. Puede crear un fichero de este tipo con: openssl dhparam -out /etc/prosody/certs/dh-2048.pem 2048

maybe-string curve [parámetro de ssl-configuration]
Curva para el protocolo Diffie-Hellman de curva elíptica. El valor predeterminado de Prosody es "secp384r1".

maybe-string-list verifyext [parámetro de ssl-configuration]
Una lista de opciones de verificación adicionales.

maybe-string password [parámetro de ssl-configuration]
Contraseña para claves privadas cifradas.
boolean c2s-require-encryption? \[parámetro de prosody-configuration\]

lista-string disable-sasl-mechanisms
Conjunto de mecanismos que no se ofrecerán nunca. Véase https://prosody.im/doc/modules/mod_saslauth. Su valor predeterminado es ‘("DIGEST-MD5")’.

boolean s2s-require-encryption?

boolean s2s-secure-auth?
Determina si el cifrado y la identificación mediante certificado son necesarias. Esto proporciona una seguridad ideal, pero necesita que los servidores con los que se comunique permitan cifrado y tengan presentes certificados válidos en los que se tenga confianza. Véase https://prosody.im/doc/s2s#security. Su valor predeterminado es ‘#f’.

lista-string s2s-insecure-domains
Muchos servidores no permiten el cifrado o tienen certificados auto-firmados. Puede proporcionar aquí una lista de dominios que no necesitarán la identificación mediante certificado. Se identificarán mediante DNS. Véase https://prosody.im/doc/s2s#security. Su valor predeterminado es ‘()’.

lista-string s2s-secure-domains
Aún en el caso de mantener s2s-secure-auth?, puede exigir certificados válidos para algunos dominios especificando una lista aquí. Véase https://prosody.im/doc/s2s#security. Su valor predeterminado es ‘()’.

string authentication
Seleciona el motor de identificación usado. La implementación predeterminada almacena las contraseñas en texto claro y usa el almacenamiento de datos configurado en Prosody para los datos de identificación. Si no confía en su servidor le recomendamos que visite https://prosody.im/doc/modules/mod_auth_internal_hashed para obtener información sobre el motor de almacenamiento tras hash. Véase también https://prosody.im/doc/authentication. Su valor predeterminado es “internal_plain”.

maybe-string log

nombre-fichero pidfile
Fichero en el que se escribirá el PID. Véase https://prosody.im/doc/modules/mod_posix. Su valor predeterminado es “/var/run/prosody/prosody.pid”.
Capítulo 8: Configuración del sistema

maybe-entero-no-negativo

http-max-content-size [parámetro de prosody-configuration]
Tamaño máximo permitido del cuerpo (body) HTTP (en bytes)

maybe-string http-external-url [parámetro de prosody-configuration]

lista-virtualhost-configuration [parámetro de prosody-configuration]

virtualhosts
Una máquina (host) en Prosody es un dominio en el que se pueden crear cuentas de usuaria. Por ejemplo, si desea que sus usuarias tengan direcciones como "juan.herrero@example.com" necesitará añadir una máquina "example.com". Todas las opciones en esta lista son efectivas únicamente en esa máquina.

Fíjese: el nombre de máquina "virtual" se usa en la configuración para evitar confusión con la máquina física en la que Prosody se encuentra instalado. Una única instancia de Prosody puede proporcionar servicio para muchos domínios, cada uno definido como una entrada de máquina virtual (VirtualHost) en la configuración de Prosody. De manera alternativa, un servidor que aloja un único dominio puede tener una única entrada VirtualHost.

Véase https://prosody.im/doc/configure#virtual_host_settings.

Los campos disponibles de virtualhost-configuration son:

string domain [parámetro de virtualhost-configuration]
Dominio en el que desea que Prosody proporcione servicio.

lista-int-component-configuration [parámetro de prosody-configuration]

int-components
Los componentes son servicios adicionales en un servidor que están disponibles a los clientes, habitualmente en un subdominio del servidor principal (como por ejemplo "micomponente.example.com"). Algunos ejemplos de componentes pueden ser los servidores de salas de conversación, los directorios de usuarias o las pasarelas a otros protocolos.

Los componentes internos se implementan con módulos específicos de Prosody. Para añadir un componente interno, simplemente rellene el campo del nombre de máquina, y el módulo que desea usar para el componente.

Véase https://prosody.im/doc/components. Su valor predeterminado es ‘()

Los campos disponibles de int-component-configuration son:
todos estos campos de prosody-configuration: admins, use-libevent?,
modules-enabled, modules-disabled, groups-file, allow-registration?,
ssl, c2s-require-encryption?, disable-sasl-mechanisms, s2s-require-
encryption?, s2s-secure-auth?, s2s-insecure-domains, s2s-secure-domains,
authentication, log, http-max-content-size, http-external-url, raw-content,
además de:

```plaintext
string hostname [parámetro de int-component-configuration]
   Nombre de máquina del componente.

string plugin [parámetro de int-component-configuration]
   Módulo que desea usar para el componente.

maybe-mod-muc-configuration [parámetro de int-component-configuration]
   mod-muc
   Multi-user chat (MUC) es el módulo de Prosody que permite la creación de
   salas de conversación/conferencias para usuarias XMPP.
   Información general sobre la configuración y el uso de las salas de conversación
   multi-usuaria puede encontrarse en la documentación "Chatrooms" (https://
   prosody.im/doc/chatrooms), que debería leer si no conoce las salas de con-
   versación de XMPP.
   Los campos disponibles de mod-muc-configuration son:

   string name [parámetro de mod-muc-configuration]
      El nombre devuelto en las respuestas de descubrimiento de servicios. Su
      valor predeterminado es ‘"Prosody Chatrooms"’.

   string-o-boolean restrict-room-creation [parámetro de mod-muc-configuration]
      Si es ‘#t’, únicamente se permitirá a las administradoras la creación de
      nuevas salas de conversación. En otro caso cualquiera puede crear una
      sala. El valor ‘"local"’ restringe la creación a usuarias en el dominio
      superior del servicio. Por ejemplo ‘usuario@example.com’ puede crear
      grupos en ‘rooms.example.com’. El valor ‘"admin"’ restringe el servicio
      a las administradoras únicamente. Su valor predeterminado es ‘#f’.

   entero-no-negativo max-history-messages [parámetro de mod-muc-configuration]
      Número máximo de mensajes históricos que se enviarán a quien se acabe
      de unir a la sala. Su valor predeterminado es ‘20’.

lista-ext-component-configuration [parámetro de prosody-configuration]
   ext-components
   Los componentes externos usan XEP-0114, el cual se implementa en la mayor parte
   de componentes independientes. Para añadir un componente externo, simplemente
   rellene el campo de nombre de máquina (hostname). Véase https://prosody.im/
   doc/components. Su valor predeterminado es ‘()’.
```
Los campos disponibles de `ext-component-configuration` son:


- `string component-secret` [parámetro de `ext-component-configuration`]
  Contraseña usada por el componente para el ingreso al sistema.

- `string hostname` [parámetro de `ext-component-configuration`]
  Nombre de máquina del componente.

- `lista-entero-no-negativo component-ports` [parámetro de `prosody-configuration`]
  Puerto o puertos en los que prosody escucha conexiones de componentes. Su valor predeterminado es `'(5347)`.

- `string component-interface` [parámetro de `prosody-configuration`]
  Interfaz en la que Prosody escucha conexiones de componentes. Su valor predeterminado es `'"127.0.0.1"`.

- `maybe-raw-content raw-content` [parámetro de `prosody-configuration`]
  Contenido que se añadirá directamente al fichero de configuración.

Puede ser que únicamente desee usar un fichero prosody.cfg.lua ya creado. En ese caso, puede proporcionar un registro `opaque-prosody-configuration` como el valor de `prosody-service-type`. Como su nombre en inglés indica, una configuración opaca no tiene gran capacidad reflexiva. Los campos disponibles de `opaque-prosody-configuration` son:

- `package prosody` [parámetro de `opaque-prosody-configuration`]
  El paquete prosody.

- `string prosody.cfg.lua` [parámetro de `opaque-prosody-configuration`]
  El contenido usado para prosody.cfg.lua.

Por ejemplo, si su prosody.cfg.lua es simplemente la cadena vacía, podría instanciar el servicio de Prosody de esta manera:

```lua
(service prosody-service-type
(opaque-prosody-configuration
(prosody.cfg.lua "")))
```

**Servicio BitlBee**

BitlBee ([https://bitlbee.org](https://bitlbee.org)) es una pasarela que proporciona una interfaz IRC a una variedad de protocolos como XMPP.

- `bitlbee-service-type` [Variable Scheme]
  Este es el tipo de servicio para el daemon de pasarela IRC BitlBee ([https://bitlbee.org](https://bitlbee.org)). Su valor es un `bitlbee-configuration` (véase a continuación).
Para que BitlBee escuche en el puerto 6667 de localhost, añada esta línea a sus servicios:

(service bitlbee-service-type)

**bitlbee-configuration**

Esta es la configuración para BitlBee, con los siguientes campos:

- **interface** (predeterminada: "127.0.0.1")
- **port** (predeterminado: 6667)
  
  Escucha en la interfaz de red correspondiente a la dirección IP especificada en `interface`, en el puerto `port`.
  
  Cuando `interface` es 127.0.0.1, únicamente se permite la conexión de clientes locales; cuando es 0.0.0.0, las conexiones pueden venir de cualquier interfaz de red.

- **bitlbee** (predeterminado: `bitlbee`)
  
  El paquete BitlBee usado.

- **plugins** (predeterminados: `( )`)
  
  Lista de paquetes de módulos usados—por ejemplo, `bitlbee-discord`.

- **extra-settings** (predeterminado: "")
  
  Fragmento de configuración añadido tal cual al fichero de configuración de BitlBee.

### Servicio Quassel

Quassel ([https://quassel-irc.org/](https://quassel-irc.org/)) es un cliente IRC distribuido, lo que significa que uno o más clientes se pueden conectar y desconectar del núcleo central.

**quassel-service-type**

Es el tipo de servicio del daemon del motor IRC de Quassel ([https://quassel-irc.org/](https://quassel-irc.org/)). Su valor es un `quassel-configuration` (véase a continuación).

**quassel-configuration**

Esta es la configuración para Quassel, con los siguientes campos:

- **quassel** (predeterminado: `quassel`)
  
  El paquete Quassel usado.

- **interface** (predeterminada: "::,0.0.0.0")
- **port** (predeterminado: 4242)
  
  Escucha en la o las interfaces de red que correspondan con las direcciones IPv4 o IPv6 delimitadas por comas especificadas en `interface`, en el puerto `port`.

- **loglevel** (predeterminado: "Info")
  
  El nivel de registro deseado. Los valores aceptados son Debug, Info, Warning y Error.
8.8.12 Servicios de telefonía

Esta sección describe cómo configurar y poner en marcha un servidor Murmur. Murmur es el servidor del paquete de voz-IP (VoIP) Mumble (https://mumble.info)

```
murmur-configuration [Tipo de datos]
    El tipo de servicio del servidor Murmur. Una ejemplo de configuración podría ser el siguiente:
    (service murmur-service-type
        (murmur-configuration
            (welcome-text
                "¡Bienvenida a este servidor Murmur que se ejecuta en Guix!")
            (cert-required? #t) ; no permite ingresos con una contraseña en texto
            (ssl-cert "/etc/letsencrypt/live/mumble.example.com/fullchain.pem"
            (ssl-key "/etc/letsencrypt/live/mumble.example.com/privkey.pem"))
    )
```

Tras reconfigurar su sistema, puede establecer manualmente la contraseña de SuperUser de Murmur con la orden que se imprime durante el proceso de activación.

Se recomienda el registro de una cuenta de usuaria normal de Mumble y la concesión de permisos de administración o moderación. Puede usar el cliente mumble para ingresar como una nueva usuaria normal, registrarse usted misma, y salir del sistema. En el siguiente paso ingrese en el sistema con el nombre SuperUser, use la contraseña de SuperUser que fue establecida con anterioridad, y conceda los permisos de administración o moderación a su usuaria de nombre creada anteriormente y cree algunos canales.

Los campos disponibles de murmur-configuration son:

```
package (predeterminado: mumble)
    Paquete que contiene bin/murmurd.

user (predeterminado: "murmur")
    Usuaria que ejecutará el servidor Murmur.

group (predeterminado: "murmur")
    Grupo de la usuaria que ejecutará el servidor murmur.

port (predeterminado: 64738)
    Puerto en el que escucha el servidor.

welcome-text (predeterminado: "")
    Mensaje de bienvenida enviado a clientes tras su conexión.

server-password (predeterminada: ")
    Contraseña que debe introducirse para poder conectarse.

max-users (predeterminados: 100)
    Número máximo de usuarias que pueden estar conectadas a la vez al servidor.

max-user-bandwidth (predeterminado: #f)
    Tráfico de voz máximo que una usuaria puede mandar por segundo.
```
database-file (predeterminado: "/var/lib/murmur/db.sqlite")
Nombre de fichero de la base de datos sqlite. La usuaria del servicio se convertirá en propietaria del directorio.

log-file (predeterminado: "/var/log/murmur/murmur.log")
Nombre de fichero del fichero de registro. La usuaria del servicio se convertirá en propietaria del directorio.

autoban-attempts (predeterminados: 10)
Número máximo de ingresos al sistema que una usuaria puede llevar a cabo en autoban-timeframe sin bloquearse su acceso durante autoban-time.

autoban-timeframe (predeterminado: 120)
Marco de tiempo del bloqueo automático en segundos.

autoban-time (predeterminado: 300)
Duración en segundos del periodo que permanecerá bloqueado un cliente cuando viole los límites de bloqueo automático.

opus-threshold (predeterminado: 100)
Porcentaje de clientes que tienen que permitir opus antes de cambiar al algoritmo de sonido opus.

channel-nesting-limit (predeterminado: 10)
Cual puede ser el nivel de recursión de los canales.

channelname-regex (predeterminado: #f)
Una cadena en forma de expresión regular Qt que deben cumplir los nombres de canal.

username-regex (predeterminado: #f)
Una cadena en forma de expresión regular Qt que deben cumplir los nombres de usuaria.

text-message-length (predeterminado: 5000)
Número máximo de bytes que una usuaria puede enviar en un mensaje de texto.

image-message-length (predeterminado: (* 128 1024))
Número máximo de bytes que una usuaria puede enviar en un mensaje de imagen.

cert-required? (predeterminado: #f)
Si su valor es #t no se aceptarán los clientes que usen identificación débil por contraseña. Las usuarias deben haber completado la herramienta de creación de certificado para unirse.

remember-channel? (predeterminado: #f)
Si murmur debe recordar el último canal en el que estaba cada usuaria cuando se desconectó y debe colocarla en el canal recordado cuando vuelva a unirse.
allow-html? (predeterminado: #f)
Si se permite html en mensajes de texto, comentarios de usuaria y descripciones de canal.

allow-ping? (predeterminado: #f)
Proporcionar un valor verdadero expone el número de usuarias actual, el número máximo de usuarias y el ancho de banda máximo del servidor por cliente a usuarias sin identificación. En el cliente Muble, esta información se muestra en el diálogo Connect/Conexión.

Desactivar esta opción impedirá la escucha pública en el servidor.

bonjour? (predeterminado: #f)
Si el servidor debe anunciarse a sí mismo en la red local a través del protocolo “bonjour”.

send-version? (predeterminado: #f)
Si se debe exponer la versión del servidor murmur en las peticiones ping.

log-days (predeterminado: 31)
Murmur también almacena registros en la base de datos, los cuales son accesibles mediante RPC. El valor predeterminado es 31 días del mes, pero puede establecer esta configuración a 0 para mantener los registros para siempre, o -1 para desactivar el registro en la base de datos.

obfuscate-ips? (predeterminado: #t)
Si las IP registradas deben ofuscarse para proteger la privacidad de las usuarias.

ssl-cert (predeterminado: #f)
Nombre del fichero del certificado SSL/TLS usado para conexiones cifradas.

(ssl-cert "/etc/letsencrypt/live/example.com/fullchain.pem")

ssl-key (predeterminada: #f)
Ruta de fichero de la clave privada de ssl usada para las conexiones cifradas.

(ssl-key "/etc/letsencrypt/live/example.com/privkey.pem")

ssl-dh-params (predeterminado: #f)
Nombre del fichero codificado con PEM con parámetros Diffie-Hellman para el cifrado SSL/TLS. De manera alternativa puede establecer su valor a "@ffdhe2048", "@ffdhe3072", "@ffdhe4096", "@ffdhe6144" o "@ffdhe8192" para usar los parámetros contenidos en el RFC 7919.

ssl-ciphers (predeterminado: #f)
La opción ssl-ciphers selecciona los protocolos de cifrado disponibles para su uso en SSL/TLS.

Esta opción se especifica mediante el uso de la notación de listas de prot. de cifrado de OpenSSL (https://www.openssl.org/docs/apps/ciphers.html#CIPHER-LIST-FORMAT).
Capítulo 8: Configuración del sistema

Se recomienda que pruebe su cadena de protocolos con 'openssl ciphers <cadena>' antes de configurarla aquí, para que compruebe que protocolos obtendrá. Tras establecer el valor de esta opción se recomienda que inspeccione el registro de Murmur para asegurarse de que Murmur usa los protocolos que esperaba que usase.

Fíjese: El cambio de esta opción puede impactar en la compatibilidad con versiones anteriores de su servidor Murmur, y puede eliminar la posibilidad de conexión desde clientes Mumble antiguos.

public-registration (predeterminado: #f)
Debe ser un registro <murmur-public-registration-configuration> o #f.

Puede registrar de manera opcional su servidor en la lista pública de servidores que el cliente mumble muestra al inicio. No puede registrar su servidor si tiene establecida una contraseña para el servidor (server-password), o establece allow-ping como #f.

Puede tomar algunas horas hasta que se muestre en la lista pública.

file (predeterminado: #f)

Forma opcional alternativa de forzar el valor de esta configuración.

murmur-public-registration-configuration

Configuración para registro público de un servicio de murmur.

name Este es el nombre mostrado de su servidor. No debe confundirse con el nombre de máquina.

password Una contraseña para identificar su registro. Las actualizaciones siguientes necesitarán la misma contraseña. No pierda su contraseña.


hostname (predeterminado: #f)

De manera predeterminada su servidor se enumerará por sus direcciones IP. Si se usa esta opción, en vez de eso se enlazará a través de este nombre de máquina.

8.8.13 Servicios de monitorización

Servicio Tailon

Tailon (https://tailon.readthedocs.io/) es una aplicación web para la visualización y búsqueda en ficheros de registro.

El ejemplo siguiente configura el servicio con los valores predeterminados. Por omisión, se puede acceder a Tailon en el puerto 8080 (http://localhost:8080).

```
(service tailon-service-type)

El ejemplo siguiente personaliza más la configuración de Tailon, añadiendo sed a la lista de órdenes permitidas.

```

```
(service tailon-service-type
tailon-configuration

```
Capítulo 8: Configuración del sistema

(config-file
  (tailon-configuration-file
    (allowed-commands '("tail" "grep" "awk" "sed"))))))

tailon-configuration

[Tipo de datos]

Tipo de datos que representa la configuración de Tailon. Este tipo tiene los siguientes parámetros:

cfg (predeterminado: (tailon-configuration-file))

El fichero de configuración usado por Tailon. Su valor puede ser un registro 'tailon-configuration-file' o cualquier expresión-G (véase Sección 6.7 [Expresiones-G], página 108).

Por ejemplo, para usar un fichero local, se puede usar la función 'local-file':

  (service tailon-service-type
    (tailon-configuration
      (config-file (local-file './mi-tailon.conf'))))

package (predeterminado: tailon)

El paquete 'tailon' usado.

tailon-configuration-file

[Tipo de datos]

Tipo de datos que representa las opciones de configuración de Tailon. Este tipo tiene los siguientes parámetros:

files (predeterminados: (list "/var/log"))

Lista de ficheros a mostrar. La lista puede incluir cadenas para un único fichero o directorio, o una lista, donde el primer elemento es el nombre de la subsección, y los elementos restantes son los ficheros o directorios de dicha subsección.

bind (predeterminado: "localhost:8080")

Dirección y puerto al que Tailon debe asociarse.

relative-root (predeterminado: #f)

Ruta URL usada por Tailon, use #f para no usar una ruta.

allow-transfers? (predeterminado: #t)

Permite la descarga de ficheros de registro en la interfaz web.

follow-names? (predeterminado: #t)

Permite la lectura de ficheros todavía no existentes.

tail-lines (predeterminado: 200)

Número de líneas a leer inicialmente de cada fichero.

allowed-commands (predeterminadas: (list "tail" "grep" "awk"))

Órdenes cuya ejecución está permitida. Por defecto, 'sed' está desactivado.

dbug? (predeterminado: #f)

Proporcione el valor #t en debug? para mostrar mensajes de depuración.
wrap-lines (predeterminado: #t)
Estado inicial del recorte de líneas en la interfaz web. El valor #t hace que las líneas se recorten inicialmente (comportamiento predeterminado), y el valor #f hace que no se recorten de manera inicial.

http-auth (predeterminado: #f)
Identificación de HTTP usada. Use #f para no permitir la identificación (el valor predeterminado). Los valores permitidos son "digest" o "basic".

users (predeterminado: #f)
Si la identificación de HTTP está activa (véase http-auth), el acceso se restringirá a las credenciales proporcionadas aquí. Para configurar usuarias, use una lista de pares, donde el primer elemento del par es el nombre de la usuaria, y el segundo elemento del par es la contraseña.

(tailon-configuration-file
 (http-auth "basic")
 (users '(("usuaria1" . "contraseña1")
          ("usuaria2" . "contraseña2"))))

Servicio Darkstat

Darkstat es un programa de interceptación de paquetes que captura el tráfico de la red, calcula estadísticas sobre su uso y proporciona informes a través de HTTP.

Variable Scheme darkstat-service-type
Este es el tipo de servicio del servicio darkstat (https://unix4lyfe.org/darkstat/), su valor debe ser un registro darkstat-configuration como en este ejemplo:

(service darkstat-service-type
 (darkstat-configuration
  (interface "eno1")))

darkstat-configuration
Tipo de datos que representa la configuración de darkstat.

package (predeterminado: darkstat)
El paquete darkstat usado.

interface
Captura el tráfico en la interfaz de red especificada.

port (predeterminado: "667")
Asocia la interfaz web al puerto especificado.

bind-address (predeterminada: "127.0.0.1")
Asocia la interfaz web a la dirección especificada.

base (predeterminada: "/")
Especifica la ruta de la URL base. Puede ser útil si darkstat se accede a través de un proxy inverso.
Servicio del exportador de nodos Prometheus

El “exportador de nodos” Prometheus pone a disposición del sistema de monitorización Prometheus las estadísticas de hardware y el sistema operativo proporcionadas por el núcleo Linux. Este servicio debe desplegarse en todos los nodos físicos y máquinas virtuales, donde la monitorización de estas estadísticas sea deseable.

**Variable Scheme** prometheus-node-exporter-service-type

Este es el tipo de servicio del servicio prometheus-node-exporter ([https://github.com/prometheus/node_exporter/](https://github.com/prometheus/node_exporter/)), su valor debe ser un registro prometheus-node-exporter-configuration como en este ejemplo:

```
(service prometheus-node-exporter-service-type
  (prometheus-node-exporter-configuration
   (web-listen-address ":9100")))
```

**prometheus-node-exporter-configuration**

Tipo de datos que representa la configuración de node_exporter.

*package (predeterminado: go-github-com-prometheus-node-exporter)*

El paquete prometheus-node-exporter usado.

*web-listen-address (predeterminada: ":9100")*

Asocia la interfaz web a la dirección especificada.

Servidor Zabbix

Zabbix proporciona métricas de monitorización, entre otras el uso de red, la carga de la CPU y el consumo de espacio en disco:

- Alto rendimiento, alta capacidad (capaz de monitorear cientos de miles de dispositivos).
- Detección automática de servidores, dispositivos de red e interfaces.
- Descubrimiento a bajo nivel, que permite el inicio automático de la monitorización de nuevos elementos, sistemas de ficheros o interfaces de red entre otros.
- Monitorización distribuida con administración web centralizada.
- Agentes nativos de alto rendimiento.
- Métricas SLA y KPI ITIL en los informes.
- Visión de alto nivel (negocio) de los recursos monitorizados a través de pantallas de consola visual y paneles definidas por la usuaria.
- Ejecución remota de ordenes a través de un proxy Zabbix.

Los campos disponibles de zabbix-server-configuration son:

*package zabbix-server*

El paquete zabbix-server.

*string user*

Usuraria que ejecutará el servidor Zabbix.

El valor predeterminado es “"zabbix"".
Capítulo 8: Configuración del sistema

grupo group [parámetro de zabbix-server-configuration]
Grupo que ejecutará el servidor Zabbix.
El valor predeterminado es "zabbix".

string db-host [parámetro de zabbix-server-configuration]
El nombre de máquina de la base de datos.
El valor predeterminado es "127.0.0.1".

string db-name [parámetro de zabbix-server-configuration]
Nombre de la base de datos.
El valor predeterminado es "zabbix".

string db-user [parámetro de zabbix-server-configuration]
Usuraria de la base de datos.
El valor predeterminado es "zabbix".

string db-password [parámetro de zabbix-server-configuration]
Contraseña de la base de datos. Por favor, en vez de esto use include-files con DBPassword=SECRET dentro de un fichero especificado.
El valor predeterminado es "".

number db-port [parámetro de zabbix-server-configuration]
Puerto de la base de datos.
El valor predeterminado es 5432.

string log-type [parámetro de zabbix-server-configuration]
Especifica donde se escriben los mensajes de registro:
- system - syslog.
- file - fichero especificado con el parámetro log-file.
- console - salida estándar.
El valor predeterminado es "".

string log-file [parámetro de zabbix-server-configuration]
Nombre del fichero de registro para el parámetro file de log-type.
El valor predeterminado es "/var/log/zabbix/server.log".

string pid-file [parámetro de zabbix-server-configuration]
Nombre del fichero de PID.
El valor predeterminado es "/var/run/zabbix/zabbix_server.pid".

string ssl-ca-location [parámetro de zabbix-server-configuration]
La localización de los ficheros de autoridades de certificación (CA) para la verificación de certificados SSL de los servidores.
El valor predeterminado es "/etc/ssl/certs/ca-certificates.crt".

string ssl-cert-location [parámetro de zabbix-server-configuration]
Localización de los certificados SSL de los clientes.
El valor predeterminado es "/etc/ssl/certs".
string extra-options [parámetro de zabbix-server-configuration]
Opciones adicionales que se añadirán al final del fichero de configuración de Zabbix.
El valor predeterminado es ‘”’.

ficheros-incluidos [parámetro de zabbix-server-configuration]
include-files
Puede incluir ficheros individuales o todos los ficheros en un directorio en el fichero de configuración.
El valor predeterminado es ‘()’.

Agente Zabbix
El agente Zabbix (zabbix-agent) obtiene información del servidor Zabbix.

Los campos disponibles de zabbix-agent-configuration son:

package zabbix-agent [parámetro de zabbix-agent-configuration]
El paquete zabbix-agent.

string user [parámetro de zabbix-agent-configuration]
Usuraria que ejecutará el agente Zabbix.
El valor predeterminado es ‘”zabbix”’.

gupo group [parámetro de zabbix-agent-configuration]
Grupo que ejecutará el agente Zabbix.
El valor predeterminado es ‘”zabbix”’.

string hostname [parámetro de zabbix-agent-configuration]
Nombre de máquina único y sensible a mayúsculas que es necesario para las comparaciones activas y debe corresponder con el nombre de máquina configurado en el servidor.
El valor predeterminado es ‘”Zabbix server”’.

string log-type [parámetro de zabbix-agent-configuration]
Especifica donde se escriben los mensajes de registro:
- system - syslog.
- file - fichero especificado con el parámetro log-file.
- console - salida estándar.
El valor predeterminado es ‘””’.

string log-file [parámetro de zabbix-agent-configuration]
Nombre del fichero de registro para el parámetro file de log-type.
El valor predeterminado es ‘”/var/log/zabbix/agent.log”’.

string pid-file [parámetro de zabbix-agent-configuration]
Nombre del fichero de PID.
El valor predeterminado es ‘”/var/run/zabbix/zabbix_agent.pid”’. 
lista server [parámetro de zabbix-agent-configuration]
Lista de direcciones IP, opcionalmente en notación CIDR, o nombres de máquina de servidores y proxy Zabbix. Se aceptarán conexiones entrantes únicamente desde las máquinas proporcionadas aquí.
El valor predeterminado es "127.0.0.1".

lista server-active [parámetro de zabbix-agent-configuration]
Lista de pares IP:puerto (o máquina:puerto) de servidores Zabbix y servidores Zabbix para las comprobaciones activas. Si no se especifica un puerto, se usa el puerto predeterminado. Si no se especifica este parámetro, las comprobaciones activas se desactivan.
El valor predeterminado es "127.0.0.1".

string extra-options [parámetro de zabbix-agent-configuration]
Opciones adicionales que se añadirán al final del fichero de configuración de Zabbix.
El valor predeterminado es "".

ficheros-incluidos [parámetro de zabbix-agent-configuration]
include-files
Puede incluir ficheros individuales o todos los ficheros en un directorio en el fichero de configuración.
El valor predeterminado es ‘()’.

Motor de visualización de Zabbix
Este servicio proporciona una interfaz Web al servidor Zabbix.
Los campos disponibles de zabbix-front-end-configuration son:

lista-nginx-server-configuration [parámetro de zabbix-front-end-configuration]
nginx
Configuración de NGINX.

string db-host [parámetro de zabbix-front-end-configuration]
El nombre de máquina de la base de datos.
El valor predeterminado es ‘"localhost"’.

number db-port [parámetro de zabbix-front-end-configuration]
Puerto de la base de datos.
El valor predeterminado es ‘5432’.

string db-name [parámetro de zabbix-front-end-configuration]
Nombre de la base de datos.
El valor predeterminado es ‘"zabbix"’.

string db-user [parámetro de zabbix-front-end-configuration]
Usuario de la base de datos.
El valor predeterminado es ‘"zabbix"’.
string db-password [parámetro de zabbix-front-end-configuration]
Contraseña de la base de datos. Por favor, en vez de esto use db-secret-file.
El valor predeterminado es ""

string db-secret-file [parámetro de zabbix-front-end-configuration]
Fichero secreto que se añadirá al final del fichero zabbix.conf.php. Este fichero contiene las credenciales usadas por el motor de visualización de Zabbix. Se espera que usted lo cree manualmente.
El valor predeterminado es ""

string zabbix-host [parámetro de zabbix-front-end-configuration]
Nombre de máquina del servidor Zabbix.
El valor predeterminado es "localhost"

number zabbix-port [parámetro de zabbix-front-end-configuration]
Puerto del servidor Zabbix.
El valor predeterminado es ‘10051’

8.8.14 Servicios Kerberos
El módulo (gnu services kerberos) proporciona servicios relacionados con el protocolo de identificación Kerberos.

Servicio Krb5

Los programas que usan una biblioteca cliente de Kerberos habitualmente esperan un fichero de configuración en la ruta /etc/krb5.conf. Este servicio genera dicho fichero desde una definición proporcionada en la declaración de sistema operativo. Esto no causa el inicio de ningún daemon.

Este servicio no crea ningún fichero “keytab”—debe crearlos explícitamente usted. Se ha comprobado que este servicio funciona con la biblioteca de cliente mit-krb5 del MIT. No se han probado otras implementaciones.

krb5-service-type [Variable Scheme]
Un tipo de servicio para clientes Kerberos 5.

Este es un ejemplo de su uso:

(service krb5-service-type
 (krb5-configuration
  (default-realm "EXAMPLE.COM")
  (allow-weak-crypto? #t)
  (realms (list
    (krb5-realm
     (name "EXAMPLE.COM")
     (admin-server "groucho.example.com")
     (kdc "karl.example.com")
    )
    (krb5-realm
     (name "ARGRX.EDU")
     (admin-server "kerb-admin.argrx.edu")
    )
   )))
Este ejemplo proporciona una configuración de cliente Kerberos 5 que:

- Reconoce dos dominios, `sean: “EXAMPLE.COM” y “ARGRX.EDU”`, los cuales tienen distintos servidores administrativos y centros de distribución de claves;
- El valor predeterminado será “EXAMPLE.COM” si no se especifica el dominio explícitamente por parte del cliente.
- Acepta servicios cuyos únicos tipos de cifrado implementados se sabe que son débiles.

Los tipos `krb5-realm` y `krb5-configuration` contienen muchos campos. Aquí se describen únicamente los más habitualmente usados. Para obtener una lista completa y una explicación detallada de cada campo, véase la documentación de `krb5.conf`.

**krb5-realm**

- **name**: Este campo es una cadena que identifica el nombre del dominio. Una convención habitual es el uso del nombre completo de DNS de su organización, convertido a mayúsculas.
- **admin-server**: Este campo es una cadena que identifica la máquina donde se ejecuta el servidor administrativo.
- **kdc**: Este campo es una cadena que identifica el centro de distribución de claves para el dominio.

**krb5-configuration**

- **allow-weak-crypto? (predeterminado: #f)**: Si esta opción es #t se aceptarán los servicios que únicamente ofrezcan algoritmos de cifrado que se conozca que son débiles.
- **default-realm (predeterminado: #f)**: Este campo debe ser una cadena que identifique el dominio predeterminado de Kerberos para los clientes. Debería proporcionar el nombre de su dominio Kerberos. Si este valor es #f, el dominio debe especificarse en cada principal de Kerberos cuando se invoquen programas como `kinit`.
- **realms**: Debe ser una lista no vacía de objetos `krb5-realm`, accesibles por los clientes. Normalmente, uno de ellos tendrá un campo `name` que corresponda con el campo `default-realm`.

**Servicio krb5 de PAM**

El servicio `pam-krb5` le permite la identificación para el ingreso al sistema y la gestión de contraseñas mediante Kerberos. Este servicio es necesario si desea que aplicaciones que permiten PAM lleven a cabo la identificación de usuarias mediante el uso de Kerberos.

**pam-krb5-service-type**

Un tipo de servicio para el módulo PAM de Kerberos 5.

**pam-krb5-configuration**

Tipo de datos que representa la configuración del módulo PAM de Kerberos 5. Este tipo tiene los siguientes parámetros:
pam-krb5 (predeterminado: pam-krb5)
El paquete pam-krb5 usado.

minimum-uid (predeterminado: 1000)
El ID de usuaria mínimo con el que se permitirán los intentos de identificación con Kerberos. El proceso de identificación de las cuentas locales con valores menores fallará de manera silenciosa.

8.8.15 Servicios LDAP
El módulo (gnu services authentication) proporciona el tipo nslcd-service-type, que puede usarse para la identificación a través de un servidor LDAP. Además de la configuración del servicio en sí, puede desear añadir ldap como servicio de nombres en el selector de servicios de nombres (NSS). Véase Sección 8.11 [Selector de servicios de nombres], página 391, para información detallada.

Aquí se encuentra una declaración simple de sistema operativo con la configuración predeterminada de nslcd-service-type y una configuración del selector de servicios de nombre que consulta en último lugar al servicios de nombres ldap:

```
(use-service-modules authentication)
(use-modules (gnu system nss))
...
(operating-system
...
(services
 (cons*
  (service nslcd-service-type)
  (service dhcp-client-service-type)
  %base-services))
 (name-service-switch
  (let ((services (list (name-service (name "db"))
                          (name-service (name "files")))
                  (name-service (name "ldap"))))
  (name-service-switch
   (inherit %mdns-host-lookup-nss)
   (password services)
   (shadow services)
   (group services)
   (netgroup services)
   (gshadow services))))
```

Los campos disponibles de nslcd-configuration son:

package nss-pam-ldapd [parámetro de nslcd-configuration]
El paquete nss-pam-ldapd usado.

maybe-number threads [parámetro de nslcd-configuration]
El número de hilos a iniciar que pueden gestionar peticiones y realizar consultas en LDAP. Cada hilo abre una conexión separada al servidor LDAP. Se inician 5 hilos de manera predeterminada.
El valor predeterminado es ‘disabled’. 
string uid  [parámetro de nslcd-configuration]
Especifica el id de usuario con el que debe ejecutarse el daemon.
El valor predeterminado es "nslcd".

string gid  [parámetro de nslcd-configuration]
Especifica el id de grupo con el que debe ejecutarse el daemon.
El valor predeterminado es "nslcd".

opción-registro log  [parámetro de nslcd-configuration]
Esta opción controla la forma en la que se realiza el registro a través de una lista que contiene ESQUEMA y NIVEL. El parámetro ESQUEMA puede ser o bien los símbolos "none" o "syslog", o la ruta absoluta de un fichero. El parámetro NIVEL es opcional y especifica el nivel de registro. El nivel de registro puede ser uno de los siguientes símbolos: "crit", "error", "warning", "notice", "info" o "debug". Se registran todos los mensajes con el nivel especificado o uno superior.
El valor predeterminado es '("/var/log/nslcd" info)'.

lista uri  [parámetro de nslcd-configuration]
La lista de URI de servidores LDAP. Normalmente, únicamente se usará el primer servidor y los siguientes se usan en caso de fallo.
El valor predeterminado es '"ldap://localhost:389/"'.

maybe-string ldap-version  [parámetro de nslcd-configuration]
La versión del protocolo LDAP usada. El valor predeterminado usa la versión máxima implementada por la biblioteca LDAP.
El valor predeterminado es 'disabled'.

maybe-string binddn  [parámetro de nslcd-configuration]
Especifica el nombre distinguido con el que enlazarse en el servidor de directorio para las búsquedas. El valor predeterminado se enlaza de forma anónima.
El valor predeterminado es 'disabled'.

maybe-string bindpw  [parámetro de nslcd-configuration]
Especifica las credenciales usadas para el enlace. Esta opción tiene utilidad únicamente cuando se usa con binddn.
El valor predeterminado es 'disabled'.

maybe-string rootpwmoddn  [parámetro de nslcd-configuration]
Especifica el nombre distinguido usado cuando la usuaria root intenta modificar la contraseña de una usuaria mediante el módulo de PAM.
El valor predeterminado es 'disabled'.

maybe-string rootpwmodpw  [parámetro de nslcd-configuration]
Especifica las credenciales con las que enlazarse si la usuaria root intenta cambiar la contraseña de una usuaria. Esta opción tiene utilidad únicamente cuando se usa con rootpwmoddn.
El valor predeterminado es 'disabled'.

Capítulo 8: Configuración del sistema

maybe-string \text{\texttt{sasl-mech}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica el mecanismo de SASL usado cuando se realice la identificación con SASL. El valor predeterminado es ‘disabled’.}

maybe-string \text{\texttt{sasl-realm}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica el dominio de SASL usado cuando se realice la identificación con SASL. El valor predeterminado es ‘disabled’.}

maybe-string \text{\texttt{sasl-authcid}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica la identidad de verificación usada cuando se realice la identificación con SASL. El valor predeterminado es ‘disabled’.}

maybe-string \text{\texttt{sasl-authzid}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica la identidad de autorización usada cuando se realice la identificación con SASL. El valor predeterminado es ‘disabled’.}

maybe-boolean \text{\texttt{sasl-canonicalize?}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Determina si el nombre de máquina del servidor LDAP debe transformarse a su forma canónica. Si se activa, la librería LDAP realizará una búsqueda inversa de nombre de máquina. De manera predeterminada, se delega en la biblioteca la decisión de realizar esta comprobación o no. El valor predeterminado es ‘disabled’.}

maybe-string \text{\texttt{krb5-ccname}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Establece el nombre para la caché de credenciales GSS-API de Kerberos. El valor predeterminado es ‘disabled’.}

string \text{\texttt{base}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{El directorio de búsqueda base. El valor predeterminado es ‘\texttt{“dc=example,dc=com”}’.}

opción-de-ámbito \text{\texttt{scope}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica el ámbito de búsqueda (subtree, oneleve, base o children). El ámbito predeterminado es subtree; el ámbito base casi nunca es útil para búsquedas del servicio de nombres; el ámbito children no está implementado en todos los servidores. El valor predeterminado es ‘(subtree)’.}

maybe-deref-option \text{\texttt{deref}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica la política para seguir las referencias de los alias. La política predeterminada es nunca seguir las referencias de los alias. El valor predeterminado es ‘disabled’.}

maybe-boolean \text{\texttt{referrals}} \quad \text{[parámetro de \texttt{nslcd-configuration}]} \quad \text{Especifica si el seguimiento automático de referencias debe activarse. El seguimiento de referencias es comportamiento predeterminado. El valor predeterminado es ‘disabled’.}
lista-asociación-entrada maps [parámetro de ns1cd-configuration]
Esta opción permite que se busquen atributos personalizados en vez de los atributos predeterminados de RFC 2307. Es una lista de asociaciones, de las que cada una consiste en el nombre de la asociación, el atributo de RFC 2307 al que corresponde y la expresión de búsqueda del atributo en la forma que esté disponible en el directorio.
El valor predeterminado es ‘()’.

lista-asociación-entrada filters [parámetro de ns1cd-configuration]
Una lista de filtros que consiste en el nombre de una asociación a la que se aplica el filtro y una expresión de filtrado de búsqueda de LDAP.
El valor predeterminado es ‘()’.

maybe-number bind-timelimit [parámetro de ns1cd-configuration]
Especifica el tiempo límite usado en segundos durante la conexión al servidor de directorio. El valor predeterminado son 10 segundos.
El valor predeterminado es ‘disabled’.

maybe-number timelimit [parámetro de ns1cd-configuration]
Especifica el tiempo límite (en segundos) durante el que se espera una respuesta del servidor LDAP. Un valor de cero, por omisión, hace que se espere de manera indefinida hasta que las búsquedas se completen.
El valor predeterminado es ‘disabled’.

maybe-number idle-timelimit [parámetro de ns1cd-configuration]
Especifica el periodo de inactividad (en segundos) tras el cual se cerrará la conexión con el servidor LDAP. El valor predeterminado no cierra las conexiones por inactividad.
El valor predeterminado es ‘disabled’.

maybe-number reconnect-sleeptime [parámetro de ns1cd-configuration]
Especifica en número de segundos que se dormirá cuando falle la conexión a todos los servidores LDAP. De manera predeterminada se espera un segundo entre el primer fallo y el primer reintento.
El valor predeterminado es ‘disabled’.

maybe-number reconnect-retrytime [parámetro de ns1cd-configuration]
Especifica el tiempo tras el cual el servidor LDAP se considera no disponible de manera permanente. Una vez se alcance este tiempo, los intentos se realizarán una vez en cada periodo de tiempo igual al especificado. El valor predeterminado es 10 segundos.
El valor predeterminado es ‘disabled’.

maybe-ssl-option ssl [parámetro de ns1cd-configuration]
Determina si se usa SSL/TLS o no (el comportamiento predeterminado es no hacerlo). Si se especifica ’start-tls, se usa StartTLS en vez de la transmisión del protocolo LDAP en crudo sobre SSL.
El valor predeterminado es ‘disabled’.
maybe-tls-reqcert-option tls-reqcert [parámetro de nslcd-configuration]
Especifica las comprobaciones que se deben realizar con un certificado proporcionado por el servidor. El significado de los valores se describe en la página de manual de ldap.conf(5).
El valor predeterminado es ‘disabled’.

maybe-string tls-cacertdir [parámetro de nslcd-configuration]
Especifica el directorio que contiene los certificados X.509 para la identificación de pares. Este parámetro se ignora si se usa GnuTLS.
El valor predeterminado es ‘disabled’.

maybe-string tls-cacertfile [parámetro de nslcd-configuration]
Especifica la ruta al certificado X.509 para la identificación de pares.
El valor predeterminado es ‘disabled’.

maybe-string tls-randfile [parámetro de nslcd-configuration]
Especifica la ruta de la fuente de entropía. Este parámetro se ignora si se usa GnuTLS.
El valor predeterminado es ‘disabled’.

maybe-string tls-ciphers [parámetro de nslcd-configuration]
Especifica como una cadena los algoritmos de cifrado usados para TLS.
El valor predeterminado es ‘disabled’.

maybe-string tls-cert [parámetro de nslcd-configuration]
Especifica la ruta al fichero que contiene el certificado local para la identificación de clientes con TLS.
El valor predeterminado es ‘disabled’.

maybe-string tls-key [parámetro de nslcd-configuration]
Especifica la ruta al fichero que contiene la clave privada para la identificación de clientes con TLS.
El valor predeterminado es ‘disabled’.

maybe-number pagesize [parámetro de nslcd-configuration]
Proporcione un valor superior a 0 para solicitar al servidor LDAP que proporcione los resultados divididos en páginas de acuerdo con el RFC2696. El valor predeterminado (0) no solicita resultados divididos en páginas.
El valor predeterminado es ‘disabled’.

maybe-ignore-users-option nss-initgroups-ignoreusers [parámetro de nslcd-configuration]
Esta opción previene las búsquedas de pertenencia a grupos a través de LDAP sobre las usuarias especificadas. De manera alternativa, se puede usar el valor ‘all-local. Con dicho valor nslcd construye al inicio una lista completa de usuarias que no se encuentren en LDAP.
El valor predeterminado es ‘disabled’.
maybe-number nss-min-uid [parámetro de nslcd-configuration]

Esta opción hace que se ignoren las usuarias de LDAP con un identificador numérico inferior al valor especificado.

El valor predeterminado es ‘disabled’.

maybe-number nss-uid-offset [parámetro de nslcd-configuration]

Esta opción especifica un desplazamiento que se añade a todos los identificadores numéricos de usuaria de LDAP. Puede usarse para evitar colisiones de identificadores con usuarias locales.

El valor predeterminado es ‘disabled’.

maybe-number nss-gid-offset [parámetro de nslcd-configuration]

Esta opción especifica un desplazamiento que se añade a todos los identificadores numéricos de grupos de LDAP. Puede usarse para evitar colisiones de identificadores con grupos locales.

El valor predeterminado es ‘disabled’.

maybe-boolean nss-nested-groups [parámetro de nslcd-configuration]

Cuando se activa esta opción, un grupo puede contener como atributo la pertenencia a otro grupo. Los miembros de grupos anidados se devuelven en el grupo superior y los grupos superiores se devuelven cuando se busquen los grupos de una usuaria específica. El valor predeterminado determina que no se realicen búsquedas adicionales para grupos anidados.

El valor predeterminado es ‘disabled’.

maybe-boolean nss-getgrent-skipmembers [parámetro de nslcd-configuration]

Cuando se activa esta opción, la lista de miembros de un grupo no se obtiene en las búsquedas de grupos. Las búsquedas que busquen los grupos de los que una usuaria es miembro continuarán funcionando de manera que probablemente a la usuaria se le asignen los grupos correctos durante el ingreso al sistema.

El valor predeterminado es ‘disabled’.

maybe-boolean nss-disable-enumeration [parámetro de nslcd-configuration]

Cuando se activa esta opción, las funciones que provocan la carga de todas las entradas usuaria/grupo del directorio no tendrán éxito al realizarlo. Esto puede reducir de forma dramática la carga del servidor LDAP cuando existe un gran número de usuarias y/o grupos. Esta opción no se recomienda para la mayoría de las configuraciones.

El valor predeterminado es ‘disabled’.

maybe-string validnames [parámetro de nslcd-configuration]

Esta opción puede usarse para especificar cómo se verifican en el sistema los nombres de usuaria y grupo. Este patrón se usa para comprobar todos los nombres de usuarias y grupos que se soliciten y proporcionen a través de LDAP.

El valor predeterminado es ‘disabled’. 
Capítulo 8: Configuración del sistema

maybe-boolean ignorecase [parámetro de nslcd-configuration]
Especifica si se realizarán las búsquedas sin diferenciar mayúsculas y minúsculas o no. Su activación puede abrir puntos vulnerables que permitan la omisión de las comprobaciones de autorización e introducir vulnerabilidades que permitan el envenenamiento de la caché de nscd, lo que puede provocar la denegación del servicio.
El valor predeterminado es ‘disabled’.

maybe-boolean pam-authc-ppolicy [parámetro de nslcd-configuration]
Esta opción determina si los controles de la política de contraseñas se solicitan y manejan desde el servidor LDAP cuando se realice la identificación de usuarias.
El valor predeterminado es ‘disabled’.

maybe-string pam-authc-search [parámetro de nslcd-configuration]
De manera predeterminada nslcd realiza una búsqueda LDAP con las credenciales de la usuaria tras la orden BIND (identificación) para asegurarse de que la opción BIND fue satisfactoria. La búsqueda predeterminada es una simple comprobación de la existencia del DN de la usuaria. Se puede especificar un filtro de búsqueda que se usará en vez de dicha búsqueda. Debe devolver al menos una entrada.
El valor predeterminado es ‘disabled’.

maybe-string pam-authz-search [parámetro de nslcd-configuration]
Esta opción permite la configuración detallada de las comprobaciones de autorización que deben realizarse. El filtro de búsqueda especificado es ejecutado, y si cualquier entrada corresponde se permite el acceso, el cual se deniega en caso contrario.
El valor predeterminado es ‘disabled’.

maybe-string pam-password-prohibit-message [parámetro de nslcd-configuration]
Si se proporciona esta opción, se denegará la modificación de contraseñas a través de pam_ldap y en vez de ello el mensaje especificado se presentará a la usuaria. El mensaje puede usarse para redirigir a la usuaria a un medio alternativo para el cambio de su contraseña.
El valor predeterminado es ‘disabled’.

lista pam-services [parámetro de nslcd-configuration]
Lista de nombres de servicio de PAM para los que la identificación de LDAP debería ser suficiente.
El valor predeterminado es ‘( )’.

8.8.16 Servicios Web

El módulo (gnu services web) proporciona el servidor HTTP Apache, el servidor web nginx y también un recubrimiento del daemon de fastcgi.

Servidor HTTP Apache

httpd-service-type [Variable Scheme]
Un ejemplo de configuración simple se proporciona a continuación.

```
(service httpd-service-type
  (httpd-configuration
    (config
      (httpd-config-file
        (server-name "www.example.com")
        (document-root "/srv/http/www.example.com")))))
```

Otros servicios también pueden extender el tipo `httpd-service-type` para añadir su contribución a la configuración.

```
(simple-service 'servidor-www.example.com httpd-service-type
  (list
    (httpd-virtualhost
     "*:80"
    (list (string-join '("ServerName "www.example.com""
                   "DocumentRoot /srv/http/www.example.com"
                   "\n")])))
```

Los detalles de los tipos de registro `httpd-configuration`, `httpd-module`, `httpd-config-file` y `httpd-virtualhost` se proporcionan a continuación.

### `httpd-configuration` [Tipo de datos]
Este tipo de datos representa la configuración del servicio `httpd`.

- **package** (predeterminado: `httpd`)
  El paquete `httpd` usado.

- **pid-file** (predeterminado: "/var/run/httpd")
  El fichero pid usado por el servicio de Shepherd.

- **config** (predeterminado: `httpd-config-file`)
  El fichero de configuración usado por el servicio `httpd`. El valor predeterminado es un registro `httpd-config-file`, pero también puede ser una expresión-G diferente que genere un fichero, por ejemplo mediante el uso de `plain-file`. Un fichero externo al almacén se puede especificar mediante una cadena.

### `httpd-module` [Tipo de datos]
Este es el tipo de datos que representa un módulo para el servicio `httpd`.

- **name**
  El nombre del módulo.

- **file**
  El fichero para el módulo. La ruta puede ser relativa al paquete `httpd` usado, la ruta absoluta de un fichero, o una expresión-G para un fichero dentro del almacén, por ejemplo (file-append mod-wsgi "/modules/mod_wsgi.so").

### `%default-httpd-modules` [Variable Scheme]
Una lista de objetos `httpd-module` predeterminados.
httpd-config-file [Tipo de datos]

Este tipo de datos representa un fichero de configuración para el servicio httpd.

módulos (predeterminados: %default-httpd-modules)

Módulos que deben cargarse. Aquí pueden añadir módulos adicionales, o que se carguen en la configuración adicional.

Por ejemplo, para manejar las peticiones de ficheros PHP, puede usar el módulo mod_proxy_fcgi de Apache junto con php-fpm-service-type:

```
(service httpd-service-type
 (httpd-configuration
  (config
   (httpd-config-file
    (modules (cons*
      (httpd-module
        (name "modulo_proxy")
        (file "modules/mod_proxy.so"))
      (httpd-module
        (name "module_proxy_fcgi")
        (file "modules/mod_proxy_fcgi.so"))
    %default-httpd-modules))

    (extra-config (list "
      <FilesMatch \.php$>
      SetHandler "proxy:unix:/var/run/php-fpm.sock|fcgi://localhost/
    </FilesMatch>")))

  (service php-fpm-service-type
   (php-fpm-configuration
    (socket "/var/run/php-fpm.sock")
    (socket-group "httpd")

 ))))

server-root (predeterminado: httpd)

El campo ServerRoot (raíz del servidor) en el fichero de configuración, cuyo valor predeterminado apunta al paquete httpd. Directivas como Include y LoadModule se interpretan como relativas a la raíz del servidor.

server-name (predeterminado: #f)

El campo ServerName (nombre del servidor) en el fichero de configuración, el cual se usa para especificar el esquema de peticiones, nombre de máquina y puerto que el servidor usa para su propia identificación.

This doesn’t need to be set in the server config, and can be specified in virtual hosts. The default is #f to not specify a ServerName.

document-root (predeterminado: "/srv/http")

La raíz (DocumentRoot) desde la que se proporcionan los archivos.

listen (predeterminado: ’("80")

La lista de valores para las directivas Listen en el fichero de configuración. El valor debe ser una lista de cadenas, donde cada cadena puede especificar el número de puerto en el que se escucha, y de manera opcional la dirección IP y el protocolo usados.
Capítulo 8: Configuración del sistema

pid-file (predeterminado: "/var/run/httpd")
El fichero de PID usado (PidFile). Debe corresponder con el valor de
pid-file proporcionado en httpd-configuration de manera que el ser-
vicio de Shepherd se configure de manera correcta.

error-log (predeterminado: "/var/log/httpd/error_log")
El fichero ErrorLog en el que el servidor registrará los errores.

user (predeterminado: "httpd")
La usuaria como la que el servidor responderá a las peticiones.

group (predeterminado: "httpd")
El grupo como el que el servidor responderá a las peticiones.

extra-config (predeterminadas: (list "TypesConfig etc/httpd/mime.types")
Una lista de cadenas y expresiones-G que se añadirán al final del fichero
de configuración.
Los valores con los que se extiende el servicio se añaden al final de esta
lista.

httpd-virtualhost
[Tipo de datos]
Este tipo de datos representa un bloque de configuración de máquina virtual del
servicio httpd.
Se deben añadir a la configuración adicional extra-config del servicio httpd-service.

(addresses-and-ports
Las direcciones y puertos de la directiva VirtualHost.

contents El contenido de la directiva VirtualHost; debe ser una lista de cadenas
y expresiones-G.

NGINX

nginx-service-type
[Variable Scheme]
Tipo de servicio para el servidor web NGinx (https://nginx.org/). El valor para
este tipo de servicio es un registro <nginx-configuration>.
Un ejemplo de configuración simple se proporciona a continuación.

(service nginx-service-type
    (nginx-configuration
        (server-blocks
            (list (nginx-server-configuration
                (server-name '("www.example.com")
                (root "/srv/http/www.example.com")))))


Además de añadiendo bloques de servidor a la configuración del servicio de manera directa, este servicio puede extenderse con otros servicios para añadir bloques de servidor, como en este ejemplo:

```
(simple-service 'mi-servidor-adicional nginx-service-type
  (list (nginx-server-configuration
         (root "/srv/http/sitio-adicional")
         (try-files (list "$uri" "$uri/index.html")))))
```

Durante su inicio, nginx no ha leído todavía su fichero de configuración, por lo que usa un fichero predeterminado para registrar los mensajes de error. Si se produce algún fallo al cargar su fichero de configuración, allí es donde se registran los mensajes de error. Tras la carga del fichero de configuración, el fichero de registro de errores predeterminado cambia al especificado allí. En nuestro caso, los mensajes de error durante el inicio se pueden encontrar en `/var/run/nginx/logs/error.log`, y tras la configuración en `/var/log/nginx/error.log`. La segunda ruta puede cambiarse con las opciones de configuración `log-directory`.

`nginx-configuration`  
[Tipo de datos]  
Este tipo de datos representa la configuración para NGinx. Alguna configuración puede llevarse a cabo a través de este y otros tipos de registro proporcionados, o de manera alternativa se puede proporcionar un fichero de configuración.

`nginx` (predeterminado: `nginx`)  
El paquete nginx usado.

`log-directory` (predeterminado: `"/var/log/nginx"`)  
Directorio en el que NGinx escribirá los ficheros de registro.

`run-directory` (predeterminado: `"/var/run/nginx"`)  
Directorio en el que NGinx crea el fichero de PID, y escribe ficheros temporales.

`server-blocks` (predeterminados: `'()'`)  
Una lista de bloques de servidor que se crearán en el fichero de configuración generado; los elementos deben ser del tipo `<nginx-server-configuration>`.

El ejemplo siguiente configura NGinx para proporcionar `www.example.com` a partir del directorio `/srv/http/www.example.com`, sin usar HTTPS.

```
(service nginx-service-type
  (nginx-configuration
   (server-blocks
    (list (nginx-server-configuration
           (server-name '("www.example.com")
           (root "/srv/http/www.example.com"))))))
```

`upstream-blocks` (predeterminados: `'()'`)  
Una lista de bloques upstream creada en el fichero de configuración generado, los elementos deben ser del tipo `<nginx-upstream-configuration>`.
La configuración de proveedores a través de `upstream-blocks` puede ser útil al combinarse con `location` en los registros `<nginx-server-configuration>`. El siguiente ejemplo crea la configuración de un servidor con una configuración de ruta, que hará de intermediario en las peticiones a la configuración de proveedores, que delegarán las peticiones en dos servidores.

```
(service
  nginx-service-type
  (nginx-configuration
    (server-blocks
      (list (nginx-server-configuration
        (server-name '("www.example.com")
        (root "/srv/http/www.example.com")
        (locations
          (list
            (nginx-location-configuration
              (uri "/ruta1"
              (body '("proxy_pass http://servidor-proxy;"))))))
        (upstream-blocks
          (list (nginx-upstream-configuration
            (name "servidor-proxy")
            (servers (list "servidor1.example.com"
                "servidor2.example.com")))))))))

file (predeterminado: #f)
Si se proporciona un fichero de configuración con `file`, se usará este, en vez de generar un fichero de configuración a partir de los parámetros `log-directory`, `run-directory`, `server-blocks` y `upstream-blocks` proporcionados. Para conseguir un funcionamiento adecuado, estos parámetros deben corresponder con el contenido de `file`, lo que asegura que los directorios se hayan creado durante la activación del servicio. Esto puede ser útil si ya dispone de un fichero de configuración, o no es posible hacer lo que necesita con el resto de opciones del registro `nginx-configuration`.

server-names-hash-bucket-size (predeterminado: #f)
Tamaño del cubo para las tablas hash de los nombres de servidor, cuyo valor predeterminado es `#f` para que se use el tamaño de la línea de caché de los procesadores.

server-names-hash-bucket-max-size (predeterminado: #f)
Tamaño máximo del cubo para las tablas hash de nombres de servidor.

modules (predeterminados: '())
Lista de módulos dinámicos de nginx cargados. Debe ser una lista de nombres de fichero de módulos cargables, como en este ejemplo:

```
(modules
  (list
    (file-append nginx-accept-language-module "\n
```
global-directives (default: '((events . ()))
Association list of global directives for the top level of the nginx configuration. Values may themselves be association lists.

(extra-content (predeterminado: "")
Contenido adicional para el bloque http. Debe ser una cadena o una expresión-G que evalué a una cadena.

nginx-server-configuration [Tipo de datos]
Tipo de datos que representa la configuración de un bloque de servidor nginx. Este tipo tiene los siguientes parámetros:

listen (predeterminadas: '("80" "443 ssl")
Cada directiva listen establece la dirección y el puerto para IP, o la ruta para un socket de dominio de UNIX sobre el que el servidor acepta peticiones. Se puede especificar tanto dirección y puerto como únicamente la dirección o únicamente el puerto. Una dirección puede ser también un nombre de máquina, por ejemplo:

'("127.0.0.1:8000" "127.0.0.1" "8000" "*:8000" "localhost:8000")

server-name (predeterminados: (list 'default))
Una lista de nombres de servidor que este servidor representa. 'default representa el servidor predeterminado para conexiones que no correspondan a otro servidor.

root (predeterminada: "/srv/http")
Raíz del sitio web que nginx proporcionará.

locations (predeterminado: '())
Una lista de registros nginx-location-configuration o nginx-named-location-configuration usados dentro de este bloque de servidor.

index (predeterminado: (list "index.html"))
Ficheros de índice buscados cuando los clientes solicitan un directorio. Si no se encuentra ninguno, Nginx enviará la lista de ficheros del directorio.

try-files (predeterminado: '())
Una lista de ficheros cuya existencia se comprueba en el orden especificado. nginx usará el primer fichero que encuentre para procesar la petición.

ssl-certificate (predeterminado: #f)
Lugar donde se encuentra el certificado para conexiones seguras. Proporcione #f si no dispone de un certificado o no desea usar HTTPS.

ssl-certificate-key (predeterminado: #f)
Lugar donde se encuentra la clave privada para conexiones seguras. Proporcione #f si no dispone de una clave o no desea usar HTTPS.
server-tokens? (predeterminado: #f)
Determina si el servidor debe añadir su configuración a las respuestas.

raw-content (predeterminado: ’())
Una lista de líneas que se añadirán literalmente al bloque del servidor.

nginx-upstream-configuration [Tipo de datos]
Tipo de datos que representa la configuración de un bloque upstream de nginx. Este tipo tiene los siguientes parámetros:

- name: Nombre para este grupo de servidores.
- servers: Especifica las direcciones de los servidores en el grupo. Las direcciones se pueden proporcionar mediante direcciones IP (por ejemplo ‘127.0.0.1’), nombres de dominio (por ejemplo ‘maquina1.example.com’) o rutas de socket de UNIX mediante el prefijo ‘unix:’. El puerto predeterminado para las direcciones IP o nombres de dominio es el 80, y se puede proporcionar un puerto de manera explícita.

nginx-location-configuration [Tipo de datos]
Tipo de datos que representa la configuración de un bloque location de nginx. Este tipo tiene los siguientes parámetros:

- uri: URI a la que corresponde este bloque de location.
- body: Cuerpo del bloque de localización “location”, especificado como una lista de cadenas. Puede contener muchas directivas de configuración. Por ejemplo, para pasar las peticiones a un grupo de servidores proveedores definido mediante el uso de un bloque nginx-upstream-configuration, se especificaría la siguiente directiva en el cuerpo ‘(list "proxy_pass http://nombre-proveedor;")’.

nginx-named-location-configuration [Tipo de datos]
Tipo de datos que representa la configuración de un bloque de localización con nombre de nginx. Los bloques de localizaciones con nombre se usan para la redirección de peticiones, y no se usan para el procesamiento regular de peticiones. Este tipo tiene los siguientes parámetros:

- name: Nombre que identifica este bloque de dirección location.
- body: Véase [cuerpo de nginx-location-configuration], página 310, como el cuerpo de los bloques de localizaciones con nombre puede usarse de manera similar al cuerpo de nginx-location-configuration. Una restricción es que el cuerpo de una localización con nombre no puede contener bloques de localizaciones.

**Caché Varnish**

Varnish es un servidor de caché rápida que se coloca entre aplicaciones web y usuarios finales. Hace de intermediario (proxy) en las peticiones de los clientes y almacena en caché las URL a las que se accede de manera que múltiples peticiones al mismo recurso únicamente creen una petición al motor.
Capítulo 8: Configuración del sistema

varnish-service-type [Variable Scheme]
Tipo de servicio para el daemon Varnish.

varnish-configuration [Tipo de datos]
Tipo de datos que representa la configuración del servicio varnish. Este tipo tiene los siguientes parámetros:

package (predeterminado: varnish)
El paquete Varnish usado.

name (predeterminado: "default")
Un nombre para esta instancia de Varnish. Varnish creará un directorio en /var/varnish con este nombre y mantendrá allí los ficheros temporales. Si el nombre comienza con una barra, se interpreta como un nombre absoluto de directorio.

Proporcione el parámetro -n a otros programas de Varnish para que se conecten a la instancia de dicho nombre, por ejemplo varnishncsa -n default.

backend (predeterminado: "localhost:8080")
Motor usado. Esta opción no tiene efecto si se usa vcl.

vcl (predeterminado: #f)
El programa VCL (lenguaje de configuración de Varnish) ejecutado. Si se proporciona #f, Varnish actuará de intermediario (proxy) de backend usando la configuración predeterminada. En otro caso debe ser un objeto “tipo-fichero” con sintaxis válida para VCL.

Por ejemplo, para mirar www.gnu.org (https://www.gnu.org) con VCL, se puede hacer algo similar a esto:

```vcl
(define %espejo-gnu
 (plain-file "gnu.vcl"
 "vcl 4.1;
 backend gnu { .host = \"www.gnu.org\"; }"))

(operating-system
 ;; ...
 (services (cons (service varnish-service-type
 (varnish-configuration
 (listen '(":80")
 (vcl %espejo-gnu)))
 %base-services)))
```

La configuración de una instancia de Varnish ya en ejecución se puede inspeccionar y cambiar mediante el uso de la orden varnishadm.

Consulte la guía de usuaria de Varnish (https://varnish-cache.org/docs/) y el libro de Varnish (https://book.varnish-software.com/4.0/) para obtener la documentación completa de Varnish y su lenguaje de configuración.

listen (predeterminada: '("localhost:80")
Lista de direcciones en las que Varnish escucha.
storage (predeterminado: '("malloc,128m"))
Lista de motores de almacenamiento que estarán disponibles en VCL.

parameters (predeterminados: '())
Lista de parámetros de tiempo de ejecución con la forma '(("parámetro" . "valor"))

extra-options (predeterminadas: '())
Parámetros adicionales a proporcional al proceso varnishd.

Patchwork
Patchwork es un sistema de seguimiento de parches. Puede recolectar parches enviados a listas de correo y mostrarlos en una interfaz web.

patchwork-service-type
Tipo de servicio para Patchwork.

El siguiente ejemplo muestra un servicio mínimo para Patchwork, para el dominio patchwork.example.com.

   (service patchwork-service-type
    (patchwork-configuration
      (domain "patchwork.example.com")
      (settings-module
        (patchwork-settings-module
          (allowed-hosts (list domain))
          (default-from-email "patchwork@patchwork.example.com")))
        (getmail-retriever-config
          (getmail-retriever-configuration
            (type "SimpleIMAPSSLRetriever")
            (server "imap.example.com")
            (port 993)
            (username "patchwork")
            (password-command
              (list (file-append coreutils "/bin/cat")
                "/etc/getmail-patchwork-imap-password"))
            (extra-parameters
              '((mailboxes . ("Parches"))))))

Existen tres registros para la configuración del servicio de Patchwork. El registro <patchwork-configuration> está relacionado con la configuración de Patchwork dentro del servicio HTTPD.

El campo settings-module dentro del registro <patchwork-configuration> puede llenarse con un registro <patchwork-settings-module>, que describe un módulo de configuración generado dentro del almacén de Guix.

En el campo database-configuration dentro del registro <patchwork-settings-module>, debe usarse <patchwork-database-configuration>.
patchwork-configuration

Tipo de datos que representa la configuración del servicio Patchwork. Este tipo tiene los siguientes parámetros:

patchwork (predeterminado: patchwork)
El paquete Patchwork usado.

domain
Dominio usado por Patchwork, se usa en el servicio HTTPD como “virtual host”.

settings-module
Módulo de configuración usado para Patchwork. Como una aplicación de Django, Patchwork se configura con un módulo Python que contiene los valores de configuración. Esto puede ser una instancia de un registro <patchwork-settings-module>, cualquier otro registro que represente la configuración en el almacén, o un directorio fuera del almacén.

static-path (predeterminada: "/static/"
Ruta bajo la cual el servicio HTTPD proporciona ficheros estáticos.

getmail-retriever-config
El valor del registro getmail-retriever-configuration usado con Patchwork. En caso de configurar Getmail con este valor, los mensajes se proporcionan a Patchwork.

patchwork-settings-module

Tipo de datos que representa un módulo de configuración de Patchwork. Algunas de estas opciones están directamente relacionadas con Patchwork, pero otras son relativas a Django, el entorno web usado Patchwork, o la biblioteca Django Rest Framework. Este tipo tiene los siguientes parámetros:

database-configuration (predeterminada:
(patchwork-database-configuration))
La configuración de la conexión a la base de datos usada para Patchwork. Véase el tipo de registro <patchwork-database-configuration> para más información.

secret-key-file (predeterminado: "/etc/patchwork/django-secret-key")
Patchwork, como una aplicación web Django, usa una clave secreta para firmar criptográficamente valores. Este fichero debe contener un valor único e impredecible.

Si este fichero no existe, el servicio de Shepherd patchwork-setup lo creará y rellenable con un valor aleatorio.

Esta configuración está relacionada con Django.

allowed-hosts
Lista de nombres de máquina válidos para este servicio Patchwork. Al menos debe incluir el dominio especificado en el registro <patchwork-configuration>.

Esta es una opción de configuración de Django.
default-from-email
La dirección de correo desde de la que Patchwork debe enviar el correo de manera predeterminada.
Esta es una opción de configuración de Patchwork.

static-url (predeterminada: #f)
La URL usada para proporcionar objetos estáticos. Puede ser parte de una URL, o una URL completa, pero debe terminar en /.
Si se usa el valor predeterminado, se usará el valor de static-path del registro <patchwork-configuration>.
Esta es una opción de configuración de Django.

admins (predeterminadas: '())
Direcciones de correo electrónico a las que enviar los detalles de los errores que ocurran. Cada valor debe ser una lista que contenga dos elementos, el nombre y la dirección de correo electrónico en dicho orden.
Esta es una opción de configuración de Django.

debug? (predeterminado: #f)
Determina si se ejecuta Patchwork en modo de depuración. Si se proporciona #t, se mostrarán mensajes de error detallados.
Esta es una opción de configuración de Django.

enable-rest-api? (predeterminado: #t)
Determina si se activa el API REST de Patchwork.
Esta es una opción de configuración de Patchwork.

enable-xmlrpc? (predeterminado: #t)
Determina si se activa el API XML RPC.
Esta es una opción de configuración de Patchwork.

force-https-links? (predeterminado: #t)
Determina si se usan enlaces HTTPS en las páginas de Patchwork.
Esta es una opción de configuración de Patchwork.

extra-settings (predeterminado: "")
Código adicional que colocar al final del módulo de configuración de Patchwork.

patchwork-database-configuration
Tipo de datos que representa la configuración de base de datos de Patchwork.

database-configuration
Tipo de datos que representa la configuración de base de datos de Patchwork.

database-configuration (predeterminado: "django.db.backends.postgresql_psycopg2")
Motor de base de datos usado.

database-configuration (predeterminado: "database-configuration")
Nombre de la base de datos usada.

user (predeterminada: "httpd")
Usuaria usada para la conexión a la base de datos.
password (predeterminada: "")
Contraseña usada para la conexión a la base de datos.

host (predeterminada: "")
Máquina usada para la conexión a la base de datos.

port (predeterminado: "")
Puerto en el que se conecta a la base de datos.

Mumi

mumi-service-type
El tipo de servicio para Mumi.

FastCGI
FastCGI es una interfaz entre la presentación (front-end) y el motor (back-end) de un servicio web. Es en cierto modo una característica antigua; los nuevos servicios web generalmente únicamente se comunican con HTTP entre ambas partes. No obstante, existe cierto número de servicios de motor como PHP o el acceso HTTP optimizado para repositorios Git que usan FastCGI, por lo que debemos incluirlo en Guix.

Para usar FastCGI debe configurar el servidor web de entrada12 (por ejemplo, nginx) para delegar un subconjunto de sus peticiones al motor fastcgi, que escucha en un puerto TCP local o en un socket de UNIX. Existe un programa de intermediación llamado fcgiwrap que se posiciona entre el proceso del motor y el servidor web. El servidor indica el programa del motor usado, proporcionando dicha información al proceso fcgiwrap.

fcgiwrap-service-type
El tipo de servicio para el proxy FastCGI fcgiwrap.

fcgiwrap-configuration
Tipo de datos que representa la configuración del servicio fcgiwrap. Este tipo tiene los siguientes parámetros:

package (predeterminado: fcgiwrap)
El paquete fcgiwrap usado.

socket (predeterminado: tcp:127.0.0.1:9000)
El socket donde el proceso fcgiwrap deba escuchar, como una cadena. Los valores adecuados para socket socket incluyen unix:/ruta/al/socket/unix, tcp:dirección.ip.con.puntos:puerto and tcp6:[dirección_ipv6]:puerto.

user (predeterminado: fcgiwrap)
group (predeterminado: fcgiwrap)
Los nombres de usuaria y grupo, como cadenas, con los que se ejecutará el proceso fcgiwrap. El servicio fastcgi se asegura, en caso de solicitar

12 NdT: Front-end en inglés.
específicamente el uso de nombres de usuario o grupo \texttt{fcgiwrap}, que la
usuario y/o grupo correspondientes se encuentren presentes en el sistema.
Es posible configurar un servicio web proporcionado por FastCGI para
que el servidor de fachada proporcione la información de identificación
HTTP al motor, y para permitir que \texttt{fcgiwrap} se ejecute en el proceso
del motor como la usuario local correspondiente. Para activar esta funcio-
nalidad en el motor, ejecute \texttt{fcgiwrap} mediante la usuario y grupo \texttt{root}.
Tenga en cuenta de que esta funcionalidad debe configurarse del mismo
modo en el servidor de fachada.

PHP-FPM (FastCGI Process Manager) es una implementación alternativa de FastCGI
en PHP con algunas características adicionales útiles para sitios de cualquier tamaño.
Estas características incluyen:

- Lanzamiento adaptativo de procesos
- Estadísticas básicas (similares a \texttt{mod\_status} de Apache)
- Gestión avanzada de procesos con parada/arranque coordinados
- Capacidad de iniciar procesos de trabajo con diferentes uid/gid/chroor/entorno y dife-
rentes \texttt{php.ini} (reemplaza a safe_mode)
- Registro a través de la salida estándar y de error
- Reinicio de emergencia en caso de destrucción accidental de la caché de opcode
- Posibilidad de subida acelerada
- Posibilidad de un "slowlog"
- Mejoras a FastCGI, como \texttt{fastcgi\_finish\_request()} - una función especial para terminar
una petición y enviar todos los datos mientras que se continua haciendo una tarea de
alto consumo de tiempo (conversión de video, procesamiento de estadísticas, etc.
... y muchas más.

\texttt{php-fpm-service-type} \hfill \text{[Variable Scheme]}
\hspace{2cm} Un tipo de servicio para \texttt{php-fpm}.

\texttt{php-fpm-configuration} \hfill \text{[Tipo de datos]}
\hspace{2cm} Tipo de datos para la configuración del servicio \texttt{php-fpm}.

\hspace{2cm} \texttt{php} (predeterminado: \texttt{php})
\hspace{2cm} El paquete \texttt{php} usado.

\hspace{2cm} \texttt{socket} (predeterminado: \texttt{(string-append "/var/run/php" (version-major
\hspace{4cm} (package-version \texttt{php})) "-fpm.sock"))}
\hspace{2cm} La dirección desde la que FastCGI acepta peticiones. Las sintaxis válidas
\hspace{2cm} son:

\hspace{4cm} \texttt{"dir.ecc.ión.ip:puerto"}
\hspace{4cm} Escucha con un socket TCP en la dirección especificada en
\hspace{4cm} un puerto específico.

\hspace{4cm} \texttt{"puerto"}
\hspace{4cm} Escucha en un socket TCP en todas las direcciones sobre un
\hspace{4cm} puerto específico.
"/ruta/a/socket/unix"
Escucha en un socket Unix.

user (predeterminada: php-fpm)
Usuaria que poseerá los procesos de trabajo de php.

group (predeterminado: php-fpm)
Grupo de los procesos de trabajo.

socket-user (predeterminado: php-fpm)
Usuaria que puede comunicarse con el socket de php-fpm.

socket-group (predeterminado: nginx)
Grupo que puede comunicarse con el socket de php-fpm.

pid-file (predeterminado: (string-append "/var/run/php" (version-major (package-version php)) "-fpm.pid"))
El identificador de proceso del proceso de php-fpm se escribe en este fichero cuando se ha iniciado el servicio.

log-file (predeterminado: (string-append "/var/log/php" (version-major (package-version php)) "-fpm.log"))
Registro del proceso maestro de php-fpm.

process-manager (predeterminado: (php-fpm-dynamic-process-manager-configuration))
Configuración detallada para el gestor de procesos php-fpm. Debe ser uno de los siguientes tipos:

<php-fpm-dynamic-process-manager-configuration>
<php-fpm-static-process-manager-configuration>
<php-fpm-on-demand-process-manager-configuration>

display-errors (predeterminado #f)
Determina si los errores y avisos de php deben enviarse a los clientes para que se muestren en sus navegadores. Esto es útil para la programación local con php, pero un riesgo para la seguridad de sitios públicos, ya que los mensajes de error pueden revelar contraseñas y datos personales.

timezone (predeterminado: #f)
Especifica el parámetro php_admin_value[date.timezone].

workers-logfile (predeterminado (string-append "/var/log/php" (version-major (package-version php)) "-fpm.www.log"))
Este fichero registrará las salidas por stderr de los procesos de trabajo de php. Puede proporcionarse #f para desactivar el registro.

file (predeterminado #f)
Sustituye opcionalmente la configuración al completo. Puede usar la función mixed-text-file o una ruta absoluta de un fichero para hacerlo.

php-fpm-dynamic-process-manager-configuration [Tipo de datos]
Tipo de datos para el gestor de procesos dynamic de php-fpm. Con el gestor de procesos dynamic, se mantienen procesos de trabajo disponibles en base a los límites configurados.
max-children (predeterminados: 5)
Número máximo de procesos de trabajo.

start-servers (predeterminados: 2)
Cuántos procesos de trabajo deben ejecutarse al inicio.

min-spare-servers (predeterminado: 1)
Cuántos procesos de trabajo deben mantenerse disponibles como mínimo.

max-spare-servers (predeterminados: 3)
Cuántos procesos de trabajo deben mantenerse disponibles como máximo.

php-fpm-static-process-manager-configuration [Tipo de datos]
Tipo de datos para el gestor de procesos static de php-fpm. Con el gestor de procesos static, se crea un número fijo de procesos de trabajo.

max-children (predeterminados: 5)
Número máximo de procesos de trabajo.

php-fpm-on-demand-process-manager-configuration [Tipo de datos]
Tipo de datos para el gestor de procesos on-demand de php-fpm. Con el gestor de procesos on-demand, se crean procesos de trabajo únicamente cuando se reciben peticiones.

max-children (predeterminados: 5)
Número máximo de procesos de trabajo.

process-idle-timeout (predeterminado: 10)
El tiempo en segundos tras el cual un proceso sin peticiones será eliminado.

nginx-php-fpm-location [#:nginx-package nginx] [Procedimiento Scheme]
[socket (string-append "/var/run/php" (version-major (package-version php)) ".fpm.sock")]
Función auxiliar para añadir php a una configuración nginx-server-configuration rápidamente.

Una configuración simple de servicios para nginx con php puede ser más o menos así:

(services (cons* (service dhcp-client-service-type)
(services nginx-service-type
(nginx-server-configuration
(server-name "example.com")
(root "/srv/http/
(locations
(list (nginx-php-location)))
(listen "80")
(ssl-certificate #f)
(ssl-certificate-key #f)))
%base-services))

El generadores de avatares de gato es un servicio simple para demostrar el uso de php-fpm en Nginx. Se usa para generar un avatar de gato desde una semilla, por ejemplo el hash de la dirección de correo de la usuaria.
cat-avatar-generator-service [#:cache-dir "/var/cache/cat-avatar-generator"] [#:package cat-avatar-generator] [#:configuration (nginx-server-configuration)]

Devuelve una configuración de nginx-server-configuration que hereda de configuration. Extiende la configuración de nginx para añadir un bloque de servidor que proporciona package, una versión de cat-avatar-generator. Durante su ejecución, cat-avatar-generator podrá usar cache-dir como su directorio de caché.

Una configuración simple para cat-avatar-generator puede ser más o menos así:

```scheme
def (services (cons* (cat-avatar-generator-service
  #:configuration (nginx-server-configuration
    (server-name '("example.com"))))
  ...)
  (%base-services))
```

Hpcguix-web

El programa hpcguix-web (https://github.com/UMCUGenetics/hpcguix-web/) es una interfaz web personalizable para buscar paquetes de Guix, diseñado inicialmente para usuarias de clusters de computación de alto rendimiento (HPC).

hpcguix-web-service-type

El tipo de servicio para hpcguix-web.

hpcguix-web-configuration

El tipo de datos para la configuración del servicio hpcguix-web.

specs Una expresión-G (véase Sección 6.7 [Expresiones-G], página 108) que especifica la configuración del servicio hpcguix-web. Los elementos principales en esta especificación son:

- title-prefix (predeterminado: "hpcguix | ")
  El prefijo del título de la página.

- guix-command (predeterminada: "guix")
  La orden guix.

- package-filter-proc (predeterminado: (const #t))
  Un procedimiento que especifica cómo filtrar los paquetes mostrados.

- package-page-extension-proc (predeterminado: (const '()))
  Paquete de extensión para hpcguix-web.

- menu (predeterminadas: '())
  Entradas adicionales en el menú de la página.

- channels (predeterminados: %default-channels)
  Lista de canales desde los que se construye la lista de paquetes (véase Sección 4.7 [Canales], página 54).
package-list-expiration (predeterminado: (* 12 3600))

El tiempo de expiración, en segundos, tras el cual la lista de paquetes se reconstruye desde las últimas instancias de los canales proporcionados.


package (predeterminado: hpcguix-web)

El paquete hpcguix-web usado.

Una declaración típica del servicio hpcguix-web es más o menos así:

```
(service hpcguix-web-service-type
 (hpcguix-web-configuration
  (specs
   #'(define site-config
      (hpcweb-configuration
       (title-prefix "Guix-HPC - ")
       (menu '(('"/about" "ABOUT"))))))

Nota: El servicio hpcguix-web actualiza periódicamente la lista de paquetes que publica obteniendo canales con Git. Para ello, necesita acceder a certificados X.509 de manera que pueda validar los servidores Git durante la comunicación con HTTPS, y asume que /etc/ssl/certs contiene dichos certificados.

Por lo tanto, asegúrese de añadir nss-certs u otro paquete de certificados en el campo packages de su configuración. Sección 8.10 [Certificados X.509], página 390, para más información sobre certificados X.509.

8.8.17 Servicios de certificados

El módulo (gnu services certbot) proporciona un servicio para la obtención automática de un certificado TLS válido de la autoridad de certificación Let's Encrypt. Estos certificados pueden usarse para proporcionar contenido de forma segura sobre HTTPS u otros protocolos basados en TLS, con el conocimiento de que el cliente podrá verificar la autenticidad del servidor.

Let’s Encrypt (https://letsencrypt.org/) proporciona la herramienta certbot para automatizar el proceso de certificación. Esta herramienta genera primero de manera segura una clave en el servidor. Una vez hecho realiza una petición a la autoridad de certificación (AC, CA en inglés) Let’s Encrypt para que firme la clave. La AC comprueba que la petición se origina en la máquina en cuestión mediante un protocolo de desafío-respuesta, esperando que el servidor proporcione su respuesta a través de HTTP. Si dicho protocolo se completa de manera satisfactoria, la AC firma la clave, resultando en un certificado. Dicho certificado es válido por un periodo de tiempo limitado y, por tanto, para continuar proporcionando servicios TLS, el servidor necesita solicitar a la AC periódicamente la renovación de su firma.

El servicio certbot automatiza este proceso: la generación inicial de la clave, la petición inicial de certificación al servicio Let’s Encrypt, la integración del desafío/respuesta en el servidor web, la escritura del certificado en disco, las renovaciones periódicas automáticas
y el despliegue de tareas asociadas con la renovación (por ejemplo la recarga de servicios y la copia de claves con diferentes permisos).

Certbot se ejecuta dos veces al día, en un minuto aleatorio dentro de la hora. No hará nada hasta que sus certificados estén pendientes de renovación o sean revocados, pero su ejecución regular propicia a su servicio la oportunidad de permanecer en línea en caso de que se produzca una revocación iniciada por Let’s Encrypt por alguna razón.

Mediante el uso de este servicio, usted acepta el acuerdo de suscripción ACME, que se puede encontrar aquí: https://acme-v01.api.letsencrypt.org/directory.

certbot-service-type

Un tipo de servicio para el cliente de Let’s Encrypt certbot. Su valor debe ser un registro certbot-configuration como en este ejemplo:

```
(define %procedimiento-de-despliegue-nginx
  (program-file
   "procedimiento-de-despliegue-nginx"
   #~(let ((pid (call-with-input-file "/var/run/nginx/pid" read)))
     (kill pid SIGHUP)))))

(service certbot-service-type
  (certbot-configuration
   (email "foo@example.net")
   (certificates
    (list
     (certificate-configuration
      (domains '("example.net" "www.example.net"))
      (deploy-hook %procedimiento-de-despliegue-nginx))
     (certificate-configuration
      (domains '("bar.example.net")))))))
```

Véase a continuación los detallles de certbot-configuration.

certbot-configuration

Tipo de datos que representa la configuración del servicio certbot. Este tipo tiene los siguientes parámetros:

package (predeterminado: certbot)

El paquete certbot usado.

webroot (predeterminado: /var/www)

Directorio desde el que se proporcionan los ficheros de desafío/respuesta de Let’s Encrypt.

certificates (predeterminados: ())

Una lista de configuraciones certificates-configuration para los cuales se generan certificados y se solicitan firmas. Cada certificado tiene un nombre (name) y varios dominios (domains).

correo electrónico (email)

Correo electrónico obligatorio usado para el registro, contacto de recuperación y notificaciones importantes de la cuenta.
Capítulo 8: Configuración del sistema

server (default: #f)
Optional URL of ACME server. Setting this overrides certbot’s default, which is the Let’s Encrypt server.

rsa-key-size (predeterminado: 2048)
Tamaño de la clave RSA.

default-location (predeterminado: véase a continuación)
La configuración nginx-location-configuration predeterminada. Debido a que certbot necesita proporcionar desafíos y respuestas, necesita ser capaz de ejecutar un servidor web. Se lleva a cabo extendiendo el servicio web nginx con una configuración nginx-server-configuration que escucha en los dominios domains en el puerto 80, y que contiene una configuración nginx-location-configuration para el subespacio de rutas URI /well-known/ usado por Let’s Encrypt. Véase Sección 8.8.16 [Servicios Web], página 303, para más información sobre estos tipos de datos de configuración de nginx.

Las peticiones a otras rutas URL se compararán contra la dirección predeterminada default-location, la cual, en caso de estar presente, se añade a todas las configuraciones nginx-server-configuration.

De manera predeterminada, la dirección predeterminada default-location emitirá una redirección http://dominio/... a https://dominio/..., lo que le permite definir qué proporcionará en su sitio web a través de https.

Proporcione #f para no emitir una dirección predeterminada.

Certificate-configuration [Tipo de datos]
Tipo de datos que representa la configuración de un certificado. Este tipo tiene los siguientes parámetros:

name (predeterminado: vea a continuación)
Este nombre se usa por Certbot para su mantenimiento interno y en las rutas de ficheros; no afecta al contenido del certificado en sí mismo. Para ver los nombres de certificados, ejecute certbot certificates.

Su valor predeterminado es el primer dominio proporcionado.

domains (predeterminado: ( ))
El primer dominio proporcionado será el sujeto del nombre común (CN) del certificado, y todos los dominios serán nombres alternativos (Subject Alternative Names) en el certificado.

challenge (predeterminado: #f)
El tipo de desafío que debe ejecutar certbot. Si se especifica #f, el valor por omisión es desafío HTTP. Si se especifica un valor, el valor por omisión es el módulo manual (véase authentication-hook, cleanup-hook y la documentación en https://certbot.eff.org/docs/using.html#hooks), y concede permiso a Let’s Encrypt para registrar la IP pública de la máquina que realiza la petición.
authentication-hook (predeterminado: #t)
Orden ejecutada en un shell una vez por cada desafío de certificado que debe contestarse. Durante su ejecución, la variable del shell $CERTBOT_DOMAIN contiene el dominio que se está validando, $CERTBOT_VALIDATION contiene la cadena de validación y $CERTBOT_TOKEN contiene el nombre de fichero del recurso solicitado cuando se realiza el desafío HTTP-01.

cleanup-hook (predeterminado: #f)
Orden ejecutada en un shell una vez por cada desafío de certificado que haya sido contestado por auth-hook. Durante su ejecución, las variables del shell disponibles en el script auth-hook todavía están disponibles, y adicionalmente $CERTBOT_AUTH_OUTPUT contendrá la salida estándar que produjo auth-hook.

deploy-hook (predeterminado: #f)
Orden ejecutada en un shell una vez por cada certificado emitido satisfactoriamente. Durante su ejecución, la variable del shell $RENEWED_LINEAGE apuntará al subdirectorio live de configuración (por ejemplo, "/etc/letsencrypt/live/example.com") que contiene las nuevas claves y certificados; la variable del shell $RENEWED_DOMAINS contendrá una lista delimitada por espacios de certificados de dominio renovados (por ejemplo, "example.com www.example.com").

Para cada configuración certificate-configuration, el certificado se almacena /etc/letsencrypt/live/name/fullchain.pem y la clave se almacena en /etc/letsencrypt/live/name/privkey.pem.

8.8.18 Servicios DNS
El módulo (gnu services dns) proporciona servicios relacionados con el sistema de nombres de dominio (DNS). Proporciona un servicio de servidor para el alojamiento de un servidor autorizado DNS para múltiples zonas, esclavo o maestro. Este servicio usa Knot DNS (https://www.knot-dns.cz/). Y también un servidor de caché y reenvío de DNS para la red local, que usa dnsmasq (http://www.thekelleys.org.uk/dnsmasq/doc.html).

Servicio Knot
Esta es una configuración de ejemplo de un servidor de autoridad para dos zonas, una maestra y otra esclava:

(define-zone-entries example.org.zone
 ;; Name TTL Class Type Data
 ("@" "IN" "A" "127.0.0.1")
 ("@" "IN" "NS" "ns")
 ("ns" "IN" "A" "127.0.0.1"))

(define master-zone
 (knot-zone-configuration
  (domain "example.org"))
(zone (zone-file
       (origin "example.org")
       (entries example.org.zone))))

(define slave-zone
  (knot-zone-configuration
   (domain "plop.org")
   (dnssec-policy "default")
   (master (list "plop-master"))))

(define plop-master
  (knot-remote-configuration
   (id "plop-master")
   (address (list "208.76.58.171")))))

(operating-system
 ;; ...
 (services (cons* (service knot-service-type
    (knot-configuration
     (remotes (list plop-master))
     (zones (list master-zone slave-zone))))
    ;; ...
    %base-services)))

knot-service-type [Variable Scheme]
Este es el tipo de datos para el servidor DNS Knot.
Knot DNS es un servidor de autoridad de DNS, lo que significa que puede servir múltiples zonas, es decir, nombres de dominio que compraría a una autoridad de registro de nombres. Este servidor no es un resolvedor, lo que significa que sólo puede resolver nombres para los que tiene autoridad. Este servidor puede configurarse para servir zonas como servidor maestro o como servidor esclavo con una granularidad al nivel de zona. Las zonas esclavas obtendrán sus datos de los servidores maestros, y las proporcionarán como un servidor de autoridad. Desde el punto de vista de un resolvedor, no hay diferencia entre servidor maestro y esclavo.

Los siguientes tipos de datos se usan para configurar el servidor DNS Knot:

knot-key-configuration [Tipo de datos]
Tipo de datos que representa una clave. Este tipo tiene los siguientes parámetros:

id (predeterminado: "")
Un identificador para que otros campos de configuración hagan referencia a esta clave. Los identificadores deben ser únicos y no deben estar vacíos.

algorithm (predeterminado: #f)

secret (predeterminado: "")
La clave secreta en sí.
knot-acl-configuration  [Tipo de datos]
Tipo de datos que representa una configuración de lista de control de acceso (ACL). Este tipo tiene los siguientes parámetros:

id (predeterminado: "")
Un identificador para que otros campos de configuración hagan referencia a esta clave. Los identificadores deben ser únicos y no deben estar vacíos.

address (predeterminada: '()')
Lista ordenada de direcciones IP, subredes o rangos de red representadas como cadenas. La búsqueda debe corresponder con alguna. El valor vacío significa que la comprobación de correspondencia de la dirección no es necesaria.

key (predeterminada: '()')
Lista ordenada de referencias a claves representadas como cadenas. La cadena debe corresponder con un ID de clave definido en knot-key-configuration. Ninguna clave significa que la comprobación de claves no es necesaria para este control de acceso (ACL).

action (predeterminada: '()')
Lista ordenada de acciones que se permiten o deniegan por este control de acceso (ACL). Los valores posibles son listas de cero o más elementos entre 'transfer, 'notify y 'update.

deny? (predeterminado: #f)
Cuando es verdadero, este ACL define restricciones. Las acciones enumeradas no se permiten. Cuando es falso, las acciones enumeradas se permiten.

zone-entry  [Tipo de datos]
Data type representing a record entry in a zone file. This type has the following parameters:

name (predeterminado: "@")
El nombre del registro. "@" hace referencia al origen de la zona. Los nombres son relativos al origen de la zona. Por ejemplo, en la zona example.org, "ns.example.org" en realidad hace referencia a ns.example.org.example.org. Los nombres que terminan en un punto se consideran absolutos, lo que significa que "ns.example.org." hace referencia a ns.example.org.

ttl (predeterminado: "")
El tiempo de vida (TTL) de este registro. Si no se proporciona, se usa el TTL predeterminado.

class (predeterminada: "IN")
La clase del registro. Actualmente Knot implementa únicamente "IN" y parcialmente "CH".
Capítulo 8: Configuración del sistema

**type** (predeterminado: "A")

El tipo del registro. Los tipos comunes incluyen A (dirección IPv4), AAAA (dirección IPv6), NS (servidor de nombres\textsuperscript{13}) y MX (pasarela de correo\textsuperscript{14}). Otros muchos tipos distintos se encuentran definidos.

**data** (predeterminados: "")

Los datos que contiene el registro. Por ejemplo, una dirección IP asociada con un registro A, o un nombre de dominio asociado con un registro NS. Recuerde que los nombres de dominio son relativos al origen a menos que terminen con punto.

**zone-file**

[Tipo de datos]

Tipo de datos que representa el contenido de un fichero de zona. Este tipo tiene los siguientes parámetros:

**entries** (predeterminadas: '()')

La lista de entradas. El registro SOA se genera automáticamente, por lo que no necesita ponerlo en la lista de entradas. Esta lista probablemente debería contener una entrada apuntando a su servidor DNS de autoridad. En vez de usar una lista de entradas directamente, puede usar `define-zone-entries` para definir un objeto que contenga la lista de entradas más fácilmente, que posteriormente puede proporcionar en el campo `entries` del fichero `zone-file`.

**origin** (predeterminado: "")

El nombre de su zona. Este parámetro no puede estar vacío.

**ns** (predeterminado: "ns")

El dominio de su servidor DNS primario de autoridad. El nombre es relativo al origen, a menos que termine en punto. Es obligatorio que este servidor DNS primario corresponda con un registro NS en la zona y que esté asociado a una dirección IP en la lista de entradas.

**mail** (predeterminado: "hostmaster")

Dirección de correo a través de la cual la gente puede contactar con usted, como propietaria de la zona. Se traduce a `<mail>@<origin>`.

**serial** (predeterminado: 1)

Número serie de la zona. Como se usa para tener constancia de los cambios tanto en servidores esclavos como en resolventes, es obligatorio que `nunca` decremente. Incrementa su valor siempre que haga cambios en su zona.

**refresh** (predeterminado: (* 2 24 3600))

La frecuencia con la que los servidores esclavos realizarán una transferencia de zona. Este valor es un número de segundos. Puede calcularse con multiplicaciones o con `(string->duration)`.

\textsuperscript{13} Name Server en inglés.

\textsuperscript{14} Mail eXchange en inglés
Capítulo 8: Configuración del sistema

retry (predeterminado: (* 15 60))
   El periodo tras el cual un servidor esclavo reintentará el contacto con su
   maestro cuando falle al intentarlo la primera vez.

expiry (predeterminado: (* 14 24 3600))
   Tiempo de vida (TTL) predeterminado de los registros. Los registros existen-
   tentes se consideran correctos durante al menos este periodo de tiempo.
   Tras este periodo, los resolviendo invalidarán su caché y comprobarán
   de nuevo que todavía exista.

nx (predeterminado: 3600)
   Tiempo de vida (TTL) de los registros inexistentes. Este retraso es habi-
   tualmente corto ya que deseará que sus nuevos dominios estén disponibles
   para cualquiera rápidamente.

knot-remote-configuration [Tipo de datos]
   Tipo de datos que representa una configuración remota. Este tipo tiene los siguientes
   parámetros:

   id (predeterminado: "")
      Un identificador para que otros campos de configuración hagan referencia
      a esta configuración remota. Los identificadores deben ser únicos y no
      deben estar vacíos.

   address (predeterminada: '()')
      Una lista ordenada de direcciones IP de destino. Las direcciones se prue-
      ban en secuencia. Opcionalmente se puede proporcionar el puerto con el
      separador @. Por ejemplo: (list "1.2.3.4" "2.3.4.5@53"). El puerto
      predeterminado es el 53.

   via (predeterminada: '()')
      Una lista ordenada de direcciones IP de fuente. Una lista vacía hará que
      Knot seleccione una fuente IP apropiada. Opcionalmente se puede pro-
      porcionar el puerto con el separador @. De manera predeterminada se
      selecciona al azar.

   key (predeterminada: #f)
      Referencia a una clave, esto es una cadena que contiene el identificador
      de una clave definida en el campo knot-key-configuration.

knot-keystore-configuration [Tipo de datos]
   Tipo de datos que representa un almacén de claves para alojar claves de dnssec. Este
   tipo tiene los siguientes parámetros:

   id (predeterminado: "")
      El identificador del almacén de claves. No debe estar vacío.

   backend (predeterminado: 'pem)
      El motor en el que se almacenan las claves. Puede ser 'pem o 'pkcs11.

   config (predeterminada: "/var/lib/knot/keys/keys")
      The configuration string of the backend. An example for the
      PKCS#11 is: "pkcs11:token=knot;pin-value=1234
Capítulo 8: Configuración del sistema

/kgnu/store/.../lib/pkcs11/libsofthsm2.so". For the pem backend, the string represents a path in the file system.

knot-policy-configuration [Tipo de datos]
Tipo de datos que representa una política de dnssec. El DNS Knot es capaz de firmar automáticamente sus zonas. Puede generar y gestionar sus claves de manera automática o usar las claves que usted genere.

Dnssec se implementa habitualmente usando dos claves: una clave para firma de claves (KSK) que se usa para firmar la segunda, y una clave para firma de zona (ZSK) que se usa para firmar la zona. Para establecer la confianza, la KSK necesita estar presente en la zona padre (habitualmente un dominio de nivel superior). Si su entidad de registro permite dnssec, debe mandarle el hash de su KSK de manera que puedan añadir un registro DS en su zona. No es automático y debe realizarse cada vez que cambie su KSK.

La política también define el tiempo de vida de las claves. Habitualmente, la ZSK puede cambiarse fácilmente y usa funciones criptográficas más débiles (usa parámetros de menor magnitud) para firmar los registros rápidamente, ya que cambian habitualmente. No obstante, la KSK requiere interacción manual con la entidad de registro, por lo que se cambia menos habitually y usa parámetros más fuertes debido a que únicamente firma un registro.

Este tipo tiene los siguientes parámetros:

id (predeterminado: "")
El identificador de la política. No debe estar vacío.

keystore (predeterminado: "default")
Referencia a un almacén de claves, es decir una cadena que contiene el identificador de un almacén de claves definido en un campo de knot-keystore-configuration. El identificador predeterminado "default" implica el uso del almacén de claves predeterminado (una base de datos kasp que se configura para este servicio).

manual? (predeterminado: #f)
Si la gestión de claves es manual o automática.

single-type-signing? (predeterminado: #f)
Cuando sea #t, usa el esquema de firma de tipo único (Single-Type Signing Scheme).

algorithm (predeterminado: "ecdsap256sha256")
Algoritmo para las claves de firma y las firmas emitidas.

ksk-size (predeterminado: 256)
La longitud de la KSK. Fíjese que este valor es correcto para el algoritmo predeterminado, pero sería inseguro para otros algoritmos.

zsk-size (predeterminado: 256)
La longitud de la ZSK. Fíjese que este valor es correcto para el algoritmo predeterminado, pero sería inseguro para otros algoritmos.
Capítulo 8: Configuración del sistema

**dnskey-ttl** (predeterminado: 'default)
El valor del tiempo de vida (TTL) de los registros DNSKEY añadidos al "apex" de la zona. El valor especial 'default significa el mismo valor que el TTL del SOA de la zona.

**zsk-lifetime** (predeterminado: (* 30 24 3600))
El periodo entre la publicación de la ZSK y el inicio del siguiente ciclo de renovación.

**propagation-delay** (predeterminado: (* 24 3600))
Retraso adicional añadido por cada paso del ciclo de renovación de clave. Este valor debe ser suficientemente alto para cubrir la propagación de datos del servidor maestro a todos los esclavos.

**rrsig-lifetime** (predeterminado: (* 14 24 3600))
Periodo de validez para las nuevas firmas emitidas.

**rrsig-refresh** (predeterminado: (* 7 24 3600))
Periodo de antelación con el que se realiza el refresco de la firma antes de una expiración de la misma.

**nsec3?** (predeterminado: #f)
Si es #t, se usa NSEC3 en vez de NSEC.

**nsec3-iterations** (predeterminado: 5)
Número de ejecuciones adicionales de la operación de hash.

**nsec3-salt-length** (predeterminado: 8)
La longitud del campo “salt” en octetos, que se añade al nombre de la propietaria original antes de ejecutar la operación de hash.

**nsec3-salt-lifetime** (predeterminado: (* 30 24 3600))
El periodo de validez de los campos “salt” que se generen.

**knot-zone-configuration**
Tipo de datos que representa una zona ofrecida por Knot. Este tipo tiene los siguientes parámetros:

**domain** (predeterminado: "")
El dominio ofrecido con esta configuración. No debe estar vacío.

**file** (predeterminado: "")
El fichero donde se almacena esta zona. Este parámetro se ignora para zonas maestras. Vacío significa la ruta predeterminada que depende del nombre del dominio.

**zone** (predeterminado: (zone-file))
El contenido del fichero de zona. Este parámetro se ignora para zonas esclavas. Debe contener un registro de fichero de zona.

**master** (predeterminado: '())
Lista de maestros remotos. Cuando está vacía, esta zona es maestra. Cuando tiene contenido, esta zona es esclava. Es una lista de identificadores remotos.
Capítulo 8: Configuración del sistema

**ddns-master** (predeterminado: #f)
Maestro principal. Cuando está vacío, apunta de manera predeterminada al primer maestro en la lista de maestros.

**notify** (predeterminado: '() )
Una lista de identificadores remotos de esclavos.

**acl** (predeterminado: '() )
Lista de identificadores acl.

**semantic-checks?** (predeterminado: #f)
Cuando es verdadero, añade más comprobaciones semánticas a la zona.

**disable-any?** (predeterminado: #f)
Cuando es verdadero, no permite consultas de tipo ANY.

**zonefile-sync** (predeterminado: 0)
El retraso entre una modificación en memoria y en disco. 0 significa sincronización inmediata.

**zonefile-load** (predeterminado: #f)
La forma en la que los contenidos del fichero de zona se aplican durante la carga de la zona. Los valores posibles son:
- #f para obtener el valor predeterminado de Knot,
- 'none para no usar el fichero de zona en absoluto,
- 'difference para calcular la diferencia entre los contenidos disponibles actualmente y los contenidos de la zona y los aplica a los contenidos actuales de la zona actual,
- 'difference-no-serial es igual que 'difference, pero ignora el código serie SOA en el fichero de zona, mientras que el servidor se hace cargo de él de manera automática.
- 'whole para cargar los contenidos de la zona del fichero de zona.

**journal-content** (predeterminado: '() )
La forma en la que se usa el diario para almacenar la zona y sus cambios. Los posibles valores son 'none para no usarlo en absoluto, 'changes para almacenar los cambios y 'all para almacenar los contenidos. #f proporciona un valor a esta opción, por lo que se usa el valor predeterminado de Knot.

**max-journal-usage** (predeterminado: #f)
Tamaño máximo del diario en disco. #f no proporciona un valor a esta opción, por lo que se usa el valor predeterminado de Knot.

**max-journal-depth** (predeterminado: #f)
Tamaño máximo de la historia. #f proporciona un valor a esta opción, por lo que se usa el valor predeterminado de Knot.

**max-zone-size** (predeterminado: #f)
Tamaño máximo del fichero de zona. Este límite se usa para transferencias entrantes y actualizaciones. #f no proporciona un valor a esta opción, por lo que se usa el valor predeterminado de Knot.
Capítulo 8: Configuración del sistema

**dnssec-policy** (predeterminado: #f)
Una referencia a un registro de knot-policy-configuration, o el nombre especial "default". Si el valor es #f, no se realiza firma dnssec en esta zona.

**serial-policy** (predeterminado: 'increment)
Una política entre 'increment y 'unixtime.

**knot-configuration**
[Tipo de datos]
Tipo de datos que representa la configuración Knot. Este tipo tiene los siguientes parámetros:

**knot** (predeterminado: knot)
El paquete Knot.

**run-directory** (predeterminado: "/var/run/knot")
El directorio de ejecución. Este directorio se usará para los ficheros de PID y de sockets.

**includes** (predeterminada: '() )
Una lista de cadenas u objetos “tipo-fichero” que denota otros ficheros que deben incluirse al inicio del fichero de configuración.

Puede usarse para gestionar secretos en un canal separado. Por ejemplo, las claves secretas pueden almacenarse en un fichero fuera de banda no gestionado por Guix, y por tanto no visible en /gnu/store—por ejemplo, puede almacenar su configuración de clave secreta en /etc/knot/secrets.conf e incluir este fichero en la lista includes.

Se puede generar una clave secreta tSIG (para nsupdate y transferencias de zona) con la orden keymgr del paquete knot. Tenga en cuenta que el paquete no se instala automáticamente con el servicio. El ejemplo siguiente muestra como generar una clave tSIG nueva:

```
keymgr -t misecreto > /etc/knot/secrets.conf
chmod 600 /etc/knot/secrets.conf
```

Tenga también en cuenta que la clave generada se llamará misecreto, de modo que ese nombre es el que debe usarse en el campo key del registro knot-acl-configuration y en otros lugares que hagan referencia a esa clave.

También puede usarse para añadir configuración no implementada por esta interfaz.

**listen-v4** (predeterminada: "0.0.0.0")
La dirección IP en la que escuchar.

**listen-v6** (predeterminada: "::")
La dirección IP en la que escuchar.

**listen-port** (predeterminado: 53)
El puerto en el que escuchar.

**keys** (predeterminada: '() )
La lista de configuraciones knot-key-configuration usadas por esta configuración.
### Servicio de resolución de Knot

**knot-resolver-service-type**

El tipo del servicio de resolución de knot, cuyo valor debe ser un objeto `knot-resolver-configuration` como en este ejemplo:

```scheme
(service knot-resolver-service-type
    (knot-resolver-configuration
        (kresd-config-file (plain-file "kresd.conf" 
        net.listen('192.168.0.1', 5353)
        user('knot-resolver', 'knot-resolver')
        modules = { 'hints > iterate', 'stats', 'predict' }
        cache.size = 100 * MB
        "')))  
```


**knot-resolver-configuration**

Tipo de datos que representa la configuración de knot-resolver.

- **package** (predeterminado: `knot-resolver`)  
  El objeto paquete de la resolución de DNS de knot.

- **kresd-config-file** (predeterminado: `%kresd.conf`)  
  Objeto “tipo-fichero” con el fichero de configuración de kresd usado, de manera predeterminada escucha en `127.0.0.1` y `::1`.

- **garbage-collection-interval** (predeterminado: 1000)  
  Número de milisegundos tras los que `kres-cache-gc` realiza una limpieza periódica de la caché.

### Servicio Dnsmasq

**dnsmasq-service-type**

Es el tipo del servicio dnsmasq, cuyo valor debe ser un objeto `dnsmasq-configuration` como en este ejemplo:

```scheme
(service dnsmasq-service-type
    (dnsmasq-configuration
        (no-resolv? #t)
        (servers '("192.168.1.1")))
```

---

**acl** (predeterminado: `()' )

La lista de configuraciones `knot-acl-configuration` usadas por esta configuración.

**remotes** (predeterminada: `()' )

La lista de configuraciones `knot-remote-configuration` usadas por esta configuración.

**zones** (predeterminada: `()' )

La lista de configuraciones `knot-zone-configuration` usadas por esta configuración.
Capítulo 8: Configuración del sistema

**dnsmasq-configuration**

Tipo de datos que representa la configuración de dnsmasq.

**package (predeterminado: dnsmasq)**

El objeto paquete del servidor dnsmasq.

**no-hosts? (predeterminado: #f)**

Cuando es verdadero, no lee los nombres de máquina en /etc/hosts.

**port (predeterminado: 53)**

El puerto sobre el que se escucha. Proporcionar el valor cero deshabilita las respuestas DNS completamente, dejando las funciones DHCP y/o TFTP únicamente.

**local-service? (predeterminado: #t)**

Acepta peticiones DNS únicamente de máquinas cuya dirección esté en una subred local, es decir, subred para la que existe una interfaz en el servidor.

**listen-addresses (predeterminadas: '()**

Escucha en las direcciones IP proporcionadas.

**resolv-file (predeterminado: "/etc/resolv.conf")**

Fichero en el que se obtienen las direcciones IP de los servidores de nombres desde los que se obtienen datos.

**no-resolv? (predeterminado: #f)**

Cuando tiene valor verdadero, no se lee resolv-file.

**servers (predeterminados: '()**

Especifica directamente la dirección IP de los servidores proveedores.

**cache-size (predeterminado: 150)**

Establece el tamaño de la caché de dnsmasq. Proporcionar el valor cero desactiva el almacenamiento en caché.

**negative-cache? (predeterminado: #t)**

Cuando es falso, desactiva la caché negativa.

---

**Servicio ddclient**

El servicio de ddclient descrito a continuación ejecuta el daemon ddclient, que se encarga de actualizar automáticamente entradas DNS para proveedores de servicio como Dyn (https://dyn.com/dns/).

El ejemplo siguiente muestra cómo instanciar el servicio con su configuración predeterminada.

```
(service ddclient-service-type)
```

Fíjese que ddclient necesita acceder a las credenciales que se almacenan en un fichero de secretos, de manera predeterminada /etc/ddclient/secrets (véase secret-file a continuación). Se espera que genere este fichero manualmente, “fuera de banda” (puede incluir este fichero en la configuración del servicio, por ejemplo mediante el uso de plain-file, pero será legible por todo el mundo a través de /gnu/store). Véase los ejemplos en el directorio share/ddclient del paquete ddclient.

Los campos disponibles de ddclient-configuration son:
package ddclient
El paquete ddclient.

integer daemon
Periodo tras el cual ddclient reintentará la comprobación de IP y de nombre de dominio.
El valor predeterminado es '300'.

boolean syslog
Usa syslog para la salida.
El valor predeterminado es '#t'

string mail
Envía por correo a la usuaria.
El valor predeterminado es "root".

string mail-failure
Envía por correo las actualizaciones fallidas a la usuaria.
El valor predeterminado es "root".

string pid
El fichero de PID de ddclient.
El valor predeterminado es "'/var/run/ddclient/ddclient.pid'".

boolean ssl
Permite el uso de SSL.
El valor predeterminado es '#t'

string user
Especifica el nombre de usuaria o ID usado para la ejecución del programa ddclient.
El valor predeterminado es "ddclient".

string group
Grupo de la usuaria que ejecutará el servidor ddclient.
El valor predeterminado es "ddclient".

string secret-file
Fichero secreto que se añadirá al final del fichero ddclient.conf. Este fichero contiene las credenciales usadas por ddclient. Se espera que usted lo cree manualmente.
El valor predeterminado es "'/etc/ddclient/secrets.conf'".

lista extra-options
Opciones adicionales a agregar al final del fichero ddclient.conf.
El valor predeterminado es '( )'.

8.8.19 Servicios VPN

El módulo (gnu services vpn) proporciona servicios relacionados con las redes privadas virtuales (VPN). Proporciona un servicio cliente para que su máquina se conecte a una VPN, y un servicio servidor para que su máquina aloje una VPN. Ambos servicios usan OpenVPN (https://openvpn.net/).

openvpn-client-service [#:config (openvpn-client-configuration)]
Devuelve un servicio que ejecuta openvpn, un daemon VPN, como cliente.

openvpn-server-service [#:config (openvpn-server-configuration)]
Devuelve un servicio que ejecuta openvpn, un daemon VPN, como servidor.
Pueden ejecutarse simultáneamente.

Los campos disponibles de openvpn-client-configuration son:

package openvpn
El paquete OpenVPN.

string pid-file
El fichero de pid de OpenVPN.
El valor predeterminado es "/var/run/openvpn/openvpn.pid".

protocolo proto
El protocolo (UDP o TCP) usado para la apertura del canal entre clientes y servidores.
El valor predeterminado es ‘udp’.

dispositivo dev
El tipo de dispositivo usado para representar la conexión VPN.
El valor predeterminado es ‘tun’.

string ca
La autoridad de certificación contra la que se comprueban las conexiones.
El valor predeterminado es ‘/etc/openvpn/ca.crt’.

string cert
El certificado de la máquina en la que se ejecuta el daemon. Debe estar firmado por la autoridad proporcionada en ca.
El valor predeterminado es ‘/etc/openvpn/client.crt’.

string key
La clave de la máquina en la que se ejecuta el daemon. Debe ser la clave cuyo certificado es cert.
El valor predeterminado es ‘/etc/openvpn/client.key’.

boolean comp-lzo?
Determina si se usa el algoritmo de compresión lzo.
El valor predeterminado es ‘#t’
boolean persist-key? [parámetro de openvpn-client-configuration]
    No vuelve a leer los ficheros de claves tras la señal SIGUSR1 o –ping-restart.
    El valor predeterminado es ‘#.t’

boolean persist-tun? [parámetro de openvpn-client-configuration]
    No cierra y reabre el dispositivo TUN/TAP o ejecuta los guiones de parada e inicio
    tras el reinicio provocado por SIGUSR1 o –ping-restart.
    El valor predeterminado es ‘#.t’

boolean fast-io? [openvpn-client-configuration parameter]
    (Experimental) Optimize TUN/TAP/UDP I/O writes by avoiding a call to
    poll/epoll/select prior to the write operation.
    El valor predeterminado es ‘#.f’

number verbosity [parámetro de openvpn-client-configuration]
    Nivel de detalle en los mensajes.
    El valor predeterminado es ‘3’.

cliente-tls-auth tls-auth [parámetro de openvpn-client-configuration]
    Añade una capa adicional de verificación HMAC sobre el canal de control TLS para
    protección contra ataques de denegación de servicio (DoS).
    El valor predeterminado es ‘#.f’

maybe-string auth-user-pass [openvpn-client-configuration parameter]
    Authenticate with server using username/password. The option is a file containing
    username/password on 2 lines. Do not use a file-like object as it would be added to
    the store and readable by any user.
    Defaults to ‘disabled’.

key-usage verify-key-usage? [parámetro de openvpn-client-configuration]
    Si se comprueba que el certificado del servidor tenga la extensión de uso de servidor.
    El valor predeterminado es ‘#.t’

bind bind? [parámetro de openvpn-client-configuration]
    Asociación a un número específico de puerto local.
    El valor predeterminado es ‘#.f’

resolv-retry resolv-retry? [parámetro de openvpn-client-configuration]
    Reintentos de resolución de la dirección del servidor.
    El valor predeterminado es ‘#.t’

lista-openvpn-remote remote [parámetro openvpn-client-configuration]
    Una lista de servidores remotos a los que conectarse.
    El valor predeterminado es ‘()’.
    Los campos disponibles de openvpn-remote-configuration son:

    string name [parámetro de openvpn-remote-configuration]
        Nombre del servidor.
        El valor predeterminado es ‘"my-server"’.
number port [parámetro de openvpn-remote-configuration]
Puerto en el que escucha el servidor.
El valor predeterminado es ‘1194’.

Los campos disponibles de openvpn-server-configuration son:

package openvpn [parámetro de openvpn-server-configuration]
El paquete OpenVPN.

string pid-file [parámetro de openvpn-server-configuration]
El fichero de pid de OpenVPN.
El valor predeterminado es ‘"/var/run/openvpn/openvpn.pid"’.

protocolo proto [parámetro de openvpn-server-configuration]
El protocolo (UDP o TCP) usado para la apertura del canal entre clientes y servidores.
El valor predeterminado es ‘udp’.

dispositivo dev [parámetro de openvpn-server-configuration]
El tipo de dispositivo usado para representar la conexión VPN.
El valor predeterminado es ‘tun’.

string ca [parámetro de openvpn-server-configuration]
La autoridad de certificación contra la que se comprueban las conexiones.
El valor predeterminado es ‘"/etc/openvpn/ca.crt"’.

string cert [parámetro de openvpn-server-configuration]
El certificado de la máquina en la que se ejecuta el daemon. Debe estar firmado por la autoridad proporcionada en ca.
El valor predeterminado es ‘"/etc/openvpn/client.crt"’.

string key [parámetro de openvpn-server-configuration]
La clave de la máquina en la que se ejecuta el daemon. Debe ser la clave cuyo certificado es cert.
El valor predeterminado es ‘"/etc/openvpn/client.key"’.

boolean comp-lzo? [parámetro de openvpn-server-configuration]
Determina si se usa el algoritmo de compresión lzo.
El valor predeterminado es ‘#t’

boolean persist-key? [parámetro de openvpn-server-configuration]
No vuelve a leer los ficheros de claves tras la señal SIGUSR1 o –ping-restart.
El valor predeterminado es ‘#t’

boolean persist-tun? [parámetro de openvpn-server-configuration]
No cierra y reabra el dispositivo TUN/TAP o ejecuta los guiones de parada e inicio tras el reinicio provocado por SIGUSR1 o –ping-restart.
El valor predeterminado es ‘#t’
boolean fast-io? [openvpn-server-configuration parameter]
(Experimental) Optimize TUN/TAP/UDP I/O writes by avoiding a call to poll/epoll/select prior to the write operation.
El valor predeterminado es ‘#f’

number verbosity [parámetro de openvpn-server-configuration]
Nivel de detalle en los mensajes.
El valor predeterminado es ‘3’.

servidor-tls-auth tls-auth [parámetro de openvpn-server-configuration]
Añade una capa adicional de verificación HMAC sobre el canal de control TLS para protección contra ataques de denegación de servicio (DoS).
El valor predeterminado es ‘#f’

number port [parámetro de openvpn-server-configuration]
Especifica el número de puerto en el que escucha el servidor.
El valor predeterminado es ‘1194’.

ip-máscara server [parámetro de openvpn-server-configuration]
Una IP y una máscara que especifiquen la subred dentro de la red virtual.
El valor predeterminado es ‘10.8.0.0 255.255.255.0’.

cidr6 server-ipv6 [parámetro de openvpn-server-configuration]
La especificación de una subred IPv6 dentro de la red virtual en notación CIDR.
El valor predeterminado es ‘#f’

string dh [parámetro de openvpn-server-configuration]
El fichero de parámetros Diffie-Hellman.
El valor predeterminado es ‘/etc/openvpn/dh2048.pem’.

string ifconfig-pool-persist [parámetro de openvpn-server-configuration]
El fichero que registra IP de clientes.
El valor predeterminado es ‘/etc/openvpn/ipp.txt’.

gateway redirect-gateway? [parámetro de openvpn-server-configuration]
Cuando sea verdadero, el servidor actuará como una pasarela para sus clientes.
El valor predeterminado es ‘#f’

boolean client-to-client? [parámetro de openvpn-server-configuration]
Cuando es verdadero, se permite la comunicación entre clientes dentro de la VPN.
El valor predeterminado es ‘#f’

keepalive keepalive [parámetro de openvpn-server-configuration]
Hace que se envíen mensajes tipo-ping en ambas direcciones a través del enlace de modo que cada extremo conozca si el otro extremo no está disponible. keepalive necesita un par. El primer elemento es el período de envío deping, y el segundo elemento es el plazo máximo antes de considerar que el otro extremo no está disponible.
Capítulo 8: Configuración del sistema

**number max-clients** [parámetro de openvpn-server-configuration]
Número máximo de clientes.
El valor predeterminado es ‘100’.

**string status** [parámetro de openvpn-server-configuration]
El fichero de estado. Este fichero muestra un pequeño informe sobre la conexión actual.
Su contenido se borra y se reescribe cada minuto.
El valor predeterminado es ‘"/var/run/openvpn/status"’.

**lista-openvpn-ccd** [parámetro de openvpn-server-configuration]

*client-config-dir*
Lista de configuración para algunos clientes.
El valor predeterminado es ‘()’.
Los campos disponibles de openvpn-ccd-configuration son:

**string name** [parámetro de openvpn-ccd-configuration]
Nombre del cliente.
El valor predeterminado es ‘"client"’.

**ip-máscara iroute** [parámetro de openvpn-ccd-configuration]
Red propia del cliente
El valor predeterminado es ‘#f’

**ip-máscara ifconfig-push** [parámetro de openvpn-ccd-configuration]
IP de la VPN del cliente.
El valor predeterminado es ‘#f’

8.8.20 Sistema de ficheros en red

El módulo (gnu services nfs) proporciona los siguientes servicios, que se usan habitualmente en relación con el montado o la exportación de árboles de directorios como sistemas de ficheros en red (NFS).

Mientras que es posible usar los componentes individuales de forma conjunta para proporcionar un servicio del sistema de ficheros en red NFS, recomendamos la configuración de un servidor NFS mediante nfs-service-type.

**Servicio NFS**

El servicio NFS se hace cargo de configurar todos los servicios de componentes de NFS, la configuración del núcleo de sistemas de ficheros e instala los ficheros de configuración en las rutas que NFS espera.

**nfs-service-type** [Variable Scheme]
Este es el tipo de datos para un servidor NFS completo.

**nfs-configuration** [Tipo de datos]
Este tipo de datos representa la configuración del servicio NFS y todos sus subsistemas.
Capítulo 8: Configuración del sistema

Tiene los siguientes parámetros:

**nfs-utils** (predeterminado: nfs-utils)
El paquete nfs-utils usado.

**nfs-version** (predeterminado: #f)
Si se proporciona una cadena como valor, el daemon rpc.nfsd se limitará a la implementación de la versión del protocolo NFS proporcionada.

**exports** (predeterminada: '() )
Una lista de directorios que el servidor NFS debe exportar. Cada entrada es una lista que consiste en dos elementos: un nombre de directorio y una cadena que contiene todas las opciones. Este es un ejemplo en el que el directorio /exportado se proporciona a todos los clientes NFS en modo de solo-lectura:

```
(nfs-configuration
  (exports
   '(("/exportado"
     "*(ro,insecure,no_subtree_check,crossmnt,fsid=0)")}))
```

**rpcmountd-port** (predeterminado: #f)
El puerto de red que el daemon rpc.mountd debe usar.

**rpcstatd-port** (predeterminado: #f)
El puerto de red que el daemon rpc.statd debe usar.

**rpcbind** (predeterminado: rpcbind)
El paquete rpcbind usado.

**idmap-domain** (predeterminado: "localdomain")
El nombre de dominio local de NFSv4.

**nfsd-port** (predeterminado: 2049)
El puerto de red que el daemon nfsd debe usar.

**nfsd-threads** (predeterminado: 8)
Número dehilos usados en el daemon nfsd.

**pipefs-directory** (predeterminado: "/var/lib/nfs/rpc_pipefs")
El directorio donde el sistema de ficheros pipefs debe montarse.

**debug** (predeterminada: '() )
Una lista de subsistemas para los cuales debe activarse la salida de depuración. Es una lista de símbolos. Cualquiera de estos símbolos son válidos: nfsd, nfs, rpc, idmap, statd o mountd.

Si no necesita un servicio NFS completo o prefiere construirlo por su cuenta puede usar los componentes individuales que se documentan a continuación.

**Servicio RPC Bind**
El servicio RPC Bind proporciona una forma de asociar números de programa con direcciones universales. Muchos servicios relacionados con NFS usan esta característica. De ahí que se inicie automáticamente cuando un servicio dependiente se inicia.
Capítulo 8: Configuración del sistema

**rpcbind-service-type**  
[Variable Scheme]  
Un tipo de servicio para el daemon de asignación de puertos RPC.

**rpcbind-configuration**  
[Tipo de datos]  
Tipo de datos que representa la configuración del servicio RPC Bind. Este tipo tiene los siguientes parámetros:

- **rpcbind** (predeterminado: `rpcbind`)  
  El paquete rpcbind usado.

- **warm-start?** (predeterminado: `#t`)  
  Si este parámetro es `#t`, el daemon leerá el fichero de estado durante el arranque, por tanto recargando la información del estado almacenada por la instancia previa.

**Pseudo-sistema de ficheros pipefs**

El sistema de ficheros pipefs se usa para transferir datos relacionados con NFS entre el núcleo y los programas de espacio de usuario.

**pipefs-service-type**  
[Variable Scheme]  
Un tipo de servicio para el pseudo-sistema de ficheros pipefs.

**pipefs-configuration**  
[Tipo de datos]  
Tipo de datos que representa la configuración del servicio del pseudo-sistema de ficheros pipefs. Este tipo tiene los siguientes parámetros:

- **mount-point** (predeterminado: `/var/lib/nfs/rpc_pipefs`)  
  El directorio al que se debe asociar el sistema de ficheros.

**Servicio del daemon GSS**

El daemon *sistema de seguridad global* (GSS) proporciona fuertes garantías de seguridad para protocolos basados en RPC. Antes de intercambiar peticiones RPC el cliente debe establecer un contexto de seguridad. Habitualmente esto se lleva a cabo con el uso de la orden `kinit` o automáticamente durante el ingreso al sistema mediante el uso de servicios PAM (véase Sección 8.8.14 [Servicios Kerberos], página 295).

**gss-service-type**  
[Variable Scheme]  
Un tipo de servicio para el daemon del sistema de seguridad global (GSS).

**gss-configuration**  
[Tipo de datos]  
Tipo de datos que representa la configuración del servicio del daemon GSS. Este tipo tiene los siguientes parámetros:

- **nfs-utils** (predeterminado: `nfs-utils`)  
  Paquete en el que se encuentra la orden `rpc.gssd`.

- **pipefs-directory** (predeterminado: `/var/lib/nfs/rpc_pipefs`)  
  El directorio donde el sistema de ficheros pipefs debe montarse.
Servicio del daemon IDMAP

El servicio del daemon idmap proporciona una asociación entre identificadores de usuario y nombres de usuario. Habitualmente es necesario para acceder sistemas de ficheros montados con NFSv4.

`idmap-service-type` [Variable Scheme]
Un tipo de servicio para el daemon de asociación de identidades (IDMAP).

`idmap-configuration` [Tipo de datos]
Tipo de datos que representa la configuración del servicio del daemon IDMAP. Este tipo tiene los siguientes parámetros:

- `nfs-utils` (predeterminado: `nfs-utils`)
  Paquete en el que se encuentra la orden `rpc.idmapd`.

- `pipefs-directory` (predeterminado: `'/var/lib/nfs/rpc_pipefs'`)
  El directorio donde el sistema de ficheros pipefs debe montarse.

- `domain` (predeterminado: `#f`)
  El nombre de dominio local de NFSv4. Debe ser una cadena o `#f`. Si es `#f` el daemon usará el nombre de dominio totalmente cualificado de la máquina.

- `verbosity` (predeterminado: `0`)
  El nivel de información de los mensajes del daemon.

8.8.21 Integración continua

Cuirass ([https://git.savannah.gnu.org/cgit/guix/guix-cuirass.git](https://git.savannah.gnu.org/cgit/guix/guix-cuirass.git)) es una herramienta de integración continua para Guix. Puede usarse tanto para el desarrollo como para proporcionar sustituciones a otras personas (véase Sección 4.3 [Sustituciones], página 44).

El módulo (gnu services cuirass) proporciona el siguiente servicio.

`cuirass-service-type` [Procedimiento Scheme]
El tipo del servicio Cuirass. Su valor debe ser un objeto `cuirass-configuration`, como se describe a continuación.

Para añadir trabajos de construcción, debe proporcionarlos en el campo `specifications` de la configuración. A continuación se encuentra un ejemplo de un servicio que consulta el repositorio de Guix y construye los paquetes de un manifiesto. Algunos de los paquetes se definen en la entrada "custom-packages", que es el equivalente de `GUIX_PACKAGE_PATH`.

```scheme
(define %especificacion-de-cuirass
  ~(list
       '(((#:name . "mi-manifiesto")
          (#:load-path-inputs . ("guix"))
          (#:package-path-inputs . ("paquetes-personalizados"))
          (#:proc-input . "guix")
          (#:proc-file . "build-aux/cuirass-gnu-system.scm")
          (#:proc . cuirass-jobs)
          (#:proc-args . ((subset . "manifests"))))
```
Mientras que la información de los trabajos de construcción se encuentra directamente en las especificaciones, la configuración global del proceso cuirass está accesible en otros campos de cuirass-configuration.

cuirass-configuration

Tipo de datos que representa la configuración de Cuirass.

log-file (predeterminado: "/var/log/cuirass.log")
Localización del fichero de registro.

web-log-file (predeterminado: "/var/log/cuirass-web.log")
Localización del fichero de registro usado por la interfaz web.

cache-directory (predeterminado: "/var/cache/cuirass")
Localización de la caché del repositorio.

user (predeterminado: "cuirass")
Propietaria del proceso cuirass.

group (predeterminado: "cuirass")
Grupo propietario del proceso cuirass.

interval (predeterminado: 60)
Número de segundos entre las consulta de repositorios seguida de los trabajos de Cuirass.

database (predeterminado: "/var/lib/cuirass/cuirass.db")
Ruta de la base de datos sqlite que contiene los resultados de construcción y las especificaciones añadidas previamente.
Capítulo 8: Configuración del sistema

344

ttl (predeterminado: (* 30 24 3600))
   Especifica el tiempo de vida (TTL) en segundos de las raíces del recolector de basura que se registran con resultados de construcciones. Esto significa que los resultados de las construcciones se protegen de la recolección de basura al menos \textit{ttl} segundos.

port (predeterminado: 8081)
   Número de puerto usado por el servidor HTTP.

host (predeterminado: "localhost")
   Escucha en la interfaz de red de la dirección \textit{host}. El comportamiento predeterminado es aceptar conexiones desde la red local.

specifications (predeterminada: `#~'()`) 
   Una expresión-G (véase Sección 6.7 [Expresiones-G], página 108) que evalúa a una lista de especificaciones, donde una especificación es una lista asociativa (véase Sección “Associations Lists” en \textit{GNU Guile Reference Manual}) cuyas claves son palabras claves (#:ejemplo-de-palabra-clave) como se muestra en el ejemplo anterior.

use-substitutes? (predeterminado: #f)
   Permite el uso de sustituciones para evitar la construcción desde las fuentes de todas las dependencias de un trabajo.

one-shot? (predeterminado: #f)
   Evalúa las especificaciones y construye las derivaciones solo una vez.

fallback? (predeterminado: #f)
   Cuando la sustitución de un binario preconstruido falle, se intentará la construcción local de los paquetes.

extra-options (predeterminadas: '())
   Extra options to pass when running the Cuirass processes.

cuirass (predeterminado: cuirass)
   El paquete Cuirass usado.

8.8.22 Servicios de gestión de energía

Daemon TLP

El módulo (\texttt{gnu services pm}) proporciona una definición de servicio Guix para la herramienta de gestión de energía de Linux TLP.

TLP activa varios modos de ahorro de energía en el núcleo y en espacio de usuario. Al contrario que \texttt{upower-service}, no es una herramienta de monitorización pasiva, puesto que aplicará una nueva configuración personalizada cada vez que se detecte una nueva fuente de energía/alimentación. Puede encontrar más información en la página de TLP (https://linrunner.de/en/tp/tlp.html).

tlp-service-type [Variable Scheme]
   El tipo de servicio para la herramienta TLP. Su valor debe ser una configuración de TLP válida (véase a continuación). Para usar la configuración predeterminada, simplemente escriba:
De manera predeterminada TLP no necesita mucha configuración, pero los parámetros de TLP se pueden ajustar mediante el uso de `tlp-configuration`.

El tipo de cada parámetro antecede la definición del mismo; por ejemplo, `boolean foo` indica que el parámetro `foo` debe especificarse como un valor lógico booleano. Los tipos que empiezan con `maybe-` indican parámetros que no aparecerán en el archivo de configuración de TLP cuando su valor sea `disabled`.

Los campos disponibles de `tlp-configuration` son:

```plaintext
package tlp

boolean tlp-enable?

string tlp-default-mode

entero-no-negativo disk-idle-secs-on-ac

entero-no-negativo disk-idle-secs-on-bat

entero-no-negativo max-lost-work-secs-on-ac

entero-no-negativo max-lost-work-secs-on-bat

maybe-lista-cadena-separada-espacios cpu-scaling-governor-on-ac
```

- `package tlp`: El paquete TLP.
- `boolean tlp-enable?`: Proporcione un valor verdadero si desea activar TLP. El valor predeterminado es `#t`.
- `string tlp-default-mode`: Modo predeterminado cuando no se puede detectar una fuente de alimentación. Las alternativas son AC (corriente alterna) y BAT (batería). El valor predeterminado es `"AC"` (corriente alterna).
- `entero-no-negativo disk-idle-secs-on-ac`: Número de segundos que el núcleo Linux debe esperar desde que el disco se queda en espera, antes de sincronizar en corriente alterna (AC). El valor predeterminado es `0`.
- `entero-no-negativo max-lost-work-secs-on-bat`: Igual que `max-lost-work-secs-on-ac` pero en modo BAT (batería). El valor predeterminado es `60`.
- `maybe-lista-cadena-separada-espacios cpu-scaling-governor-on-ac`: Gobernador de escalado de frecuencia del procesador en modo de corriente alterna (AC). Con el controlador intel_pstate, las alternativas son “powersave” (ahorro de...

El valor predeterminado es ‘disabled’.

**maybe-lista-cadena-separada-espacios** [parámetro de tlp-configuration]

cpu-scaling-governor-on-bat

Igual que max-lost-work-secs-on-ac pero en modo BAT (batería).

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-scaling-min-freq-on-ac

Establece la frecuencia mínima disponible para el controlador de escalado en AC.

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-scaling-max-freq-on-ac

Establece la frecuencia máxima disponible para el controlador de escalado en AC.

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-scaling-min-freq-on-bat

Establece la frecuencia mínima disponible para el controlador de escalado en BAT.

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-scaling-max-freq-on-bat

Establece la frecuencia máxima disponible para el controlador de escalado en BAT.

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-min-perf-on-ac

Limita el estado-P mínimo para controlar la disipación de potencia del procesador en modo AC. Los valores se indican como un porcentaje de rendimiento disponible.

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-max-perf-on-ac

Limita el estado-P máximo para controlar la disipación de potencia del procesador en modo AC. Los valores se indican como un porcentaje de rendimiento disponible.

El valor predeterminado es ‘disabled’.

**maybe-entero-no-negativo** [parámetro de tlp-configuration]

cpu-min-perf-on-bat

Igual que cpu-min-perf-on-ac pero en modo BAT (batería).

El valor predeterminado es ‘disabled’.
maybe-entero-no-negativo  
**cpu-max-perf-on-bat**  
Igual que **cpu-max-perf-on-ac** pero en modo BAT (batería).  
El valor predeterminado es ‘disabled’.

maybe-boolean **cpu-boost-on-ac**?  
Activa la característica “turbo boost” del procesador en modo AC (corriente alterna).  
El valor predeterminado es ‘disabled’.

maybe-boolean **cpu-boost-on-bat**?  
Igual que **cpu-boost-on-ac** pero en modo BAT (batería).  
El valor predeterminado es ‘disabled’.

**boolean sched-powersave-on-ac**?  
Permite al núcleo Linux minimizar el número de núcleos/hyper-thread del procesador usados bajo condiciones de baja carga.  
El valor predeterminado es ‘#f’

**boolean sched-powersave-on-bat**?  
Igual que **sched-powersave-on-ac**? pero en modo BAT (batería).  
El valor predeterminado es ‘#t’

**boolean nmi-watchdog**?  
Activa el proceso guardián (watchdog) NMI del núcleo Linux.  
El valor predeterminado es ‘#f’

**maybe-string phc-controls**  
Para núcleos Linux con el parche PHC aplicado, cambia los voltajes del procesador.  
Un valor de ejemplo sería ‘"F:V F:V F:V F:V:V"’.  
El valor predeterminado es ‘disabled’.

**string energy-perf-policy-on-ac**  
Establece la política de alto rendimiento del procesador en vez del ahorro de energía mientras durante el uso de corriente alterna (AC). Las alternativas son “performance” (alto rendimiento), “normal” y “powersave” (ahorro de energía).  
El valor predeterminado es ‘"performance"’.

**string energy-perf-policy-on-bat**  
Igual que **energy-perf-policy-ac** pero en modo BAT (batería).  
El valor predeterminado es ‘"powersave"’.

**lista-cadena-separada-espacios disks-devices**  
Dispositivos de disco duro.

**lista-cadena-separada-espacios disk-apm-level-on-ac**  
Nivel de APM (gestión avanzada de energía) del disco duro.
lista-cadena-separada-espacios [parámetro de tlp-configuration]
disk-apm-level-on-bat
Igual que disk-apm-bat pero en modo BAT (batería).

maybe-lista-cadena-separada-espacios [parámetro de tlp-configuration]
disk-spindown-timeout-on-ac
Plazo para la parada rotacional del disco duro. Se debe especificar un valor por cada disco duro declarado.
El valor predeterminado es ‘disabled’.

maybe-lista-cadena-separada-espacios [parámetro de tlp-configuration]
disk-spindown-timeout-on-bat
Igual que disk-spindown-timeout-on-ac pero en modo BAT (batería).
El valor predeterminado es ‘disabled’.

maybe-lista-cadena-separada-espacios [parámetro de tlp-configuration]
disk-iosched
Selecciona el planificador de E/S para dispositivos de disco. Se debe especificar un valor por cada disco duro declarado. Ejemplos de alternativas son “cfq”, “deadline” y “noop”.
El valor predeterminado es ‘disabled’.

string sata-linkpwr-on-ac [parámetro de tlp-configuration]
Nivel de gestión agresiva de energía del enlace (ALPM) de SATA. Las alternativas son “min_power” (energía mínima), “medium_power” (energía media) y “max_performance” (máximo rendimiento).
El valor predeterminado es ‘”max_performance”’.

string sata-linkpwr-on-bat [parámetro de tlp-configuration]
Igual que sata-linkpwr-ac pero en modo BAT (batería).
El valor predeterminado es ‘”min_power”’.

maybe-string sata-linkpwr-blacklist [parámetro de tlp-configuration]
Excluye los dispositivos SATA especificados de la gestión de energía del enlace.
El valor predeterminado es ‘disabled’.

maybe-on-off-boolean [parámetro de tlp-configuration]
ahci-runtime-pm-on-ac?
Activa la gestión de energía de tiempo de ejecución para controladores AHCI y discos en modo AC.
El valor predeterminado es ‘disabled’.

maybe-on-off-boolean [parámetro de tlp-configuration]
ahci-runtime-pm-on-bat?
Igual que ahci-runtime-pm-on-ac pero en modo BAT (batería).
El valor predeterminado es ‘disabled’.
### Capítulo 8: Configuración del sistema

**entero-no-negativo**  
[parámetro de tlp-configuration]

**ahci-runtime-pm-timeout**  
Segundos de inactividad antes de suspender el disco.  
El valor predeterminado es ‘15’.

**string pcie-aspm-on-ac**  
[parámetro de tlp-configuration]

Nivel de gestión de energía de estado activo de PCI Express. Las alternativas son “default” (predeterminado), “performance” (rendimiento) y “powersave” (ahorro de energía).  
El valor predeterminado es ‘”performance”’.

**string pcie-aspm-on-bat**  
[parámetro de tlp-configuration]

Igual que pcie-aspm-ac pero en modo BAT (batería).  
El valor predeterminado es ‘”powersave”’.

**string radeon-power-profile-on-ac**  
[parámetro de tlp-configuration]

Nivel de velocidad de reloj de gráficos Radeon. Las alternativas son “low” (bajo), “mid” (medio), “high” (alto), “auto” (automático) y “default” (predeterminado).  
El valor predeterminado es ‘”high”’.

**string radeon-power-profile-on-bat**  
[parámetro de tlp-configuration]

Igual que radeon-power-ac pero en modo BAT (batería).  
El valor predeterminado es ‘”low”’.

**string radeon-dpm-state-on-ac**  
[parámetro de tlp-configuration]

Método de gestión de energía dinámica (DPM) de Radeon. Las alternativas son “battery” (batería) y “performance” (rendimiento).  
El valor predeterminado es ‘”performance”’.

**string radeon-dpm-state-on-bat**  
[parámetro de tlp-configuration]

Igual que radeon-dpm-state-ac pero en modo BAT (batería).  
El valor predeterminado es ‘”battery”’.

**string radeon-dpm-perf-level-on-ac**  
[parámetro de tlp-configuration]

Nivel de rendimiento del DPM de Radeon. Las alternativas son “auto” (automático), “low” (bajo) y “high” (alto).  
El valor predeterminado es ‘”auto”’.

**string radeon-dpm-perf-level-on-bat**  
[parámetro de tlp-configuration]

Igual que radeon-dpm-perf-ac pero en modo BAT (batería).  
El valor predeterminado es ‘”auto”’.

**on-off-boolean wifi-pwr-on-ac?**  
[parámetro de tlp-configuration]

Modo de ahorro de energía de Wifi.  
El valor predeterminado es ‘#f’

**on-off-boolean wifi-pwr-on-bat?**  
[parámetro de tlp-configuration]

Igual que wifi-power-ac? pero en modo BAT (batería).  
El valor predeterminado es ‘#t’
y-n-boolean wol-disable? [parámetro de tlp-configuration]
Desactiva el encendido desde la red local (wake on LAN).
El valor predeterminado es ‘#t’

entero-no-negativo [parámetro de tlp-configuration]
sound-power-save-on-ac
Duración en segundos del plazo antes de activar el ahorro de energía de audio en dispositivos Intel HDA y AC97. El valor 0 desactiva el ahorro de energía.
El valor predeterminado es ‘0’.

entero-no-negativo [parámetro de tlp-configuration]
sound-power-save-on-bat
Igual que sound-powersave-ac pero en modo BAT (batería).
El valor predeterminado es ‘1’.

y-n-boolean [parámetro de tlp-configuration]
sound-power-save-controller?
Desactiva el controlador en modo de ahorro de energía en dispositivos Intel HDA.
El valor predeterminado es ‘#t’

boolean bay-poweroff-on-bat?
Activa las unidades ópticas en UltraBay/MediaBay en modo BAT. La unidad puede volver a alimentarse liberando (y reinsertando) la palanca de eyección o presionando el botón de eyección de disco en modelos más modernos.
El valor predeterminado es ‘#f’

string bay-device [parámetro de tlp-configuration]
Nombre de la unidad de dispositivos ópticos a apagar.
El valor predeterminado es ‘”sr0”’.

string runtime-pm-on-ac [parámetro de tlp-configuration]
Gestión de energía en tiempo de ejecución para dispositivos de bus PCI(e). Las alternativas son “on” y “auto”.
El valor predeterminado es ‘”on”’.

string runtime-pm-on-bat [parámetro de tlp-configuration]
Igual que runtime-pm-ac pero en modo BAT (batería).
El valor predeterminado es ‘”auto”’.

boolean runtime-pm-all? [parámetro de tlp-configuration]
Gestión de energía en tiempo de ejecución (Runtime Power Management) para todos los dispositivos del bus PCI(e), excepto los excluidos.
El valor predeterminado es ‘#t’

maybe-lista-cadena-separada-espacios [parámetro de tlp-configuration]
runtime-pm-blacklist
Excluye las direcciones de dispositivo PCI(e) especificadas de la gestión de energía en tiempo de ejecución (Runtime Power Management).
El valor predeterminado es ‘disabled’.
Capítulo 8: Configuración del sistema

**lista-cadena-separada-espacios** [parámetro de tlp-configuration]

*runtime-pm-driver-blacklist*

Excluye los dispositivos PCI(e) asignados a los controladores especificados de la gestión de energía en tiempo de ejecución (Runtime Power Management).

**boolean usb-autosuspend?** [parámetro de tlp-configuration]

Permite la suspensión automática de USB.

El valor predeterminado es ‘#t’

**maybe-string usb-blacklist** [parámetro de tlp-configuration]

Excluye los dispositivos especificados de la suspensión automática de USB.

El valor predeterminado es ‘disabled’.

**boolean usb-blacklist-wwan?** [parámetro de tlp-configuration]

Excluye los dispositivos WWAN de la suspensión automática de USB.

El valor predeterminado es ‘#t’

**maybe-string usb-whitelist** [parámetro de tlp-configuration]

Incluye los dispositivos especificados en la suspensión automática de USB, incluso cuando están excluidos por el controlador o a través de `usb-blacklist-wwan?`.

El valor predeterminado es ‘disabled’.

**maybe-boolean usb-autosuspend-disable-on-shutdown?** [parámetro de tlp-configuration]

Activa la suspensión automática de USB antes del apagado.

El valor predeterminado es ‘disabled’.

**boolean restore-device-state-on-startup?** [parámetro de tlp-configuration]

Restablece el estado de los dispositivos de radio (bluetooth, wifi, wwan) previo al apagado durante el arranque del sistema.

El valor predeterminado es ‘#f’

**Daemon Thermald**

El módulo (gnu services pm) proporciona una interfaz con thermald, un servicio de escalamiento de frecuencia de la CPU que ayuda a prevenir el sobrecalentamiento.

**thermald-service-type** [Variable Scheme]

Este es el tipo de servicio para thermald (https://01.org/linux-thermal-daemon/), el daemon Thermal de Linux, que es responsable del control del estado térmico de los procesadores y la prevención del sobrecalentamiento.

**thermald-configuration** [Tipo de datos]

Tipo de datos que representa la configuración de `thermald-service-type`.

**ignore-cpuid-check?** (predeterminado: #f)

Ignora la comprobación de cpuid durante la comprobación de procesadores permitidos.

**thermald** (predeterminado: `thermald`)

El objeto paquete de thermald.
8.8.23 Servicios de audio

El módulo (gnu services audio) proporciona un servicio para iniciar MPD (el daemon de reproducción de música).

Daemon de reproducción de música (MPD)

El daemon de reproducción de música (MPD) es un servicio que puede reproducir música mientras se controla desde la máquina local o sobre una red por una multitud de clientes.

El siguiente ejemplo muestra como se puede ejecutar mpd como "rober" en el puerto 6666. Usa pulseaudio para su salida.

```
(service mpd-service-type
   (mpd-configuration
    (user "rober")
    (port "6666"))

mpd-service-type
   El tipo de servicio para mpd.

mpd-configuration
   [Variable Scheme]
   Tipo de datos que representa la configuración de mpd.

user (predeterminada: "mpd")
   Usuraria que ejecuta mpd.

music-dir (predeterminado: "~/Music")
   El directorio para buscar ficheros de música.

playlist-dir (predeterminado: "~/.mpd/playlists")
   El directorio para almacenar listas de reproducción.

db-file (predeterminado: "~/.mpd/tag_cache")
   La localización de la base de datos de música.

state-file (predeterminado: "~/.mpd/state")
   La localización del fichero que almacena el estado actual de MPD.

sticker-file (predeterminado: "~/.mpd/sticker.sql")
   La localización de la base de datos de pegatinas.

port (predeterminado: "6600")
   Puerto sobre el que se ejecuta mpd.

address (predeterminada: "any")
   Dirección a la que mpd se asociará. Para usar un socket de dominio de Unix puede especificarse una ruta absoluta.

outputs (predeterminadas: "(list (mpd-output))")
  Las salidas de audio que MPD puede usar. De manera predeterminada es una salida de audio única usando pulseaudio.

mpd-output
   [Tipo de datos]
   Tipo de datos que representa una salida de audio de mpd.

name (predeterminado: "MPD")
   Nombre de la salida de audio.
type (predeterminado: "pulse")
Tipo de la salida de audio.

enabled? (predeterminado: #t)
Especifica si esta salida de audio se activa cuando se inicia MPD. De manera predeterminada se activan todas las salidas de audio. Esta es la configuración predeterminada cuando no existe un fichero de estado; con un fichero de estado se restaura el estado anterior.

tags? (predeterminado: #t)
Si se proporciona el valor #f MPD no envía etiquetas a esta salida. Es útil únicamente para módulos de salida que pueden recibir etiquetas, por ejemplo el módulo de salida httpd.

always-on? (predeterminado: #f)
Si se proporciona el valor #t MPD intenta mantener abierta siempre esta salida de audio. Puede ser útil para servidores de retransmisión, donde no desea que todos los clientes se desconecten incluso cuando la reproducción se pare accidentalmente.

mixer-type
Este campo acepta un símbolo que especifica que mezclador debe usarse para esta salida de audio: el mezclador hardware, el mezclador software, el mezclador null (permite manejar el volumen pero sin ningún efecto; puede usarse para implementar un mezclador externo) o ningún mezclador (none).

extra-options (predeterminadas: '(())
Una lista asociativa de símbolos de opción con valores de cadenas que se añadirán al final de la configuración de las salidas de audio.

El ejemplo siguiente muestra una configuración de mpd que proporciona una salida de retransmisión de audio por HTTP.

```
(service mpd-service-type
  (mpd-configuration
   (outputs
    (list (mpd-output
      (name "streaming")
      (type "httpd")
      (mixer-type 'null)
      (extra-options
       ('((encoder . "vorbis")
         (port . "8080"))))))))
```

### 8.8.24 Servicios de virtualización

El módulo (gnu services virtualization) proporciona servicios para los daemon libvirt y virtlog, así como otros servicios relacionados con la virtualización.
Daemon de Libvirt

`libvirtd` es un daemon del que se compone el lado del servidor del sistema de gestión de virtualización libvirt. Este daemon se ejecuta en las máquinas servidoras y realiza las tareas de gestión necesarias para los sistemas virtualizados.

`libvirt-service-type` [Variable Scheme]

```
(service libvirt-service-type
  (libvirt-configuration
    (unix-sock-group "libvirt")
    (tls-port "16555")))
```

Los campos disponibles de `libvirt-configuration` son:

`package libvirt` [parámetro de `libvirt-configuration`]
Paquete libvirt.

`boolean listen-tls?` [parámetro de `libvirt-configuration`]
Opción para la escucha de conexiones seguras TLS en el puerto TCP/IP público. Debe haberse proporcionado valor a `listen` para que tenga algún efecto.
Es necesario configurar una autoridad de certificación (CA) y emitir certificados de servidor antes de usar esta característica.
El valor predeterminado es ‘#t’

`boolean listen-tcp?` [parámetro de `libvirt-configuration`]
Escucha de conexiones TCP sin cifrar en el puerto TCP/IP público. Debe haberse proporcionado valor a `listen` para que tenga algún efecto.
El uso del socket TCP necesita de manera predeterminada identificación SASL. Únicamente se permiten mecanismos SASL que implementen cifrado de datos. Estos son DIGEST_MDI5 y GSSAPI (Kerberos5).
El valor predeterminado es ‘#f’

`string tls-port` [parámetro de `libvirt-configuration`]
Puerto en el que se aceptan conexiones seguras. Puede ser un número de puerto o un nombre de servicio.
El valor predeterminado es ‘"16514"’.

`string tcp-port` [parámetro de `libvirt-configuration`]
Puerto en el que se aceptan conexiones inseguras. Puede ser un número de puerto o un nombre de servicio.
El valor predeterminado es ‘"16509"’.

`string listen-addr` [parámetro de `libvirt-configuration`]
Dirección IP o nombre de máquina usado para las conexiones de clientes.
El valor predeterminado es ‘"0.0.0.0"’. 
boolean mdns-adv?  [parámetro de libvirt-configuration]
   Opción que determina el anuncio mDNS del servicio libvirt.
   De manera alternativa puede desactivarse para todos los servicios en una máquina
   parando el daemon Avahi.
   El valor predeterminado es ‘#f’

string mdns-name  [parámetro de libvirt-configuration]
   Nombre predeterminado del anuncio mDNS. Debe ser único en la red de distribución
   inmediata.
   El valor predeterminado es ‘"Virtualization Host <hostname>"’.<

string unix-sock-group  [parámetro de libvirt-configuration]
   Grupo propietario del socket de dominio de UNIX. Puede usarse para permitir a
   un conjunto de usuarias “de confianza” acceso a las funcionalidades de gestión sin
   convertirse en root.
   El valor predeterminado es ‘"root"’..

string unix-sock-ro-perms  [parámetro de libvirt-configuration]
   Permisos del socket UNIX de sólo lectura. Se usa únicamente para monitorizar el
   estado de las máquinas virtuales.
   El valor predeterminado es ‘"0777"’.

string unix-sock-rw-perms  [parámetro de libvirt-configuration]
   Permisos del socket UNIX de lectura/escritura. El valor predeterminado únicamente
   permite acceso a root. Si PolicyKit se encuentra activo en el socket, el valor prede-
   terminado cambiará para permitir acceso universal (es decir, 0777).
   El valor predeterminado es ‘"0770"’.

string unix-sock-admin-perms  [parámetro de libvirt-configuration]
   Permisos del socket UNIX de administración. El valor predeterminado únicamente
   permite acceso a la propietaria (root), no lo cambie a menos que esté completamente
   segura de a quién expone el acceso.
   El valor predeterminado es ‘"0777"’.

string unix-sock-dir  [parámetro de libvirt-configuration]
   Directorio en el que los sockets se encuentran/crean.
   El valor predeterminado es ‘"/var/run/libvirt"’.

string auth-unix-ro  [parámetro de libvirt-configuration]
   Esquema de identificación para los sockets de solo-lectura de UNIX. Los permisos
   predeterminados del socket permiten la conexión de cualquier usuaria.
   El valor predeterminado es ‘"polkit"’.

15 R/O: Read-Only en inglés.
16 R/W: Read-Write en inglés.
string auth-unix-rw  
Esquema de identificación para los sockets de lectura/escritura de UNIX. Los permisos predeterminados del socket permiten la conexión únicamente a root. Si se activó en la compilación de libvirt la interoperabilidad con PolicyKit, el valor predeterminado es usar la identificación “policykit”.
El valor predeterminado es "policykit".

string auth-tcp  
Esquema de identificación para los sockets TCP. Si no activa SASL, todo el tráfico TCP estará en texto plano. No lo haga más allá de un escenario de desarrollo/pruebas.
El valor predeterminado es "sasl".

string auth-tls  
Esquema de identificación para los sockets TLS. Los sockets TLS ya se encuentran cifrados gracias a la capa TLS, y una identificación limitada se realiza con los certificados.
También es posible hacer uso de cualquier mecanismo de identificación SASL proporcionando “sasl” en esta opción.
El valor predeterminado es "none".

lista-opcional access-drivers  
Esquema de la API de control de acceso.
De manera predeterminada una usuaria identificada puede acceder a todas las API. Los controladores de acceso pueden incluir restricciones de acceso sobre ello.
El valor predeterminado es ‘()’.

string key-file  
Ruta del fichero con la clave del servidor. Si se proporciona una cadena vacía, no se carga ninguna clave privada.
El valor predeterminado es ‘”’.

string cert-file  
Ruta del fichero con la clave del servidor. Si se proporciona una cadena vacía, no se carga ningún certificado.
El valor predeterminado es ‘”’.

string ca-file  
Ruta del fichero con la clave del servidor. Si se proporciona una cadena vacía, no se carga ningún certificado de CA.
El valor predeterminado es ‘”’.

string crl-file  
Ruta de la lista de revocaciones de certificado. Si se proporciona una cadena vacía, no se carga ninguna lista.
El valor predeterminado es ‘”’.
boolean tls-no-sanity-cert  [parámetro de libvirt-configuration]
Desactiva la verificación de los propios certificados del servidor.
Cuando libvirtd arranca, realiza algunas comprobaciones básicas sobre sus propios
certificados.
El valor predeterminado es ‘#f’

boolean tls-no-verify-cert  [parámetro de libvirt-configuration]
Desactiva la verificación de certificados de clientes.
La verificación de certificados de cliente es el mecanismo primario de identificación.
Se rechazará cualquier cliente que no presente un certificado firmado por la autoridad
de certificación (CA).
El valor predeterminado es ‘#f’

lista-opcional tls-allowed-dn-list  [parámetro de libvirt-configuration]
Lista de nombres distinguidos (DN) x509 permitidos.
El valor predeterminado es ‘()’.

lista-opcional sasl-allowed-usernames  [parámetro de libvirt-configuration]
Lista de nombres de usuaria SASL permitidos. El formato para el nombre de la usuaria
dependen del mecanismo de identificación SASL.
El valor predeterminado es ‘()’.

string tls-priority  [parámetro de libvirt-configuration]
Cambia el valor de la cadena de prioridad de TLS predeterminada en tiempo de
compilación. El valor predeterminado habitualmente es "NORMAL" a menos que se
cambie en tiempo de compilación. Proporcione este valor únicamente si desea que
libvirt se desvíe de la configuración global predeterminada.
El valor predeterminado es "NORMAL".

integer max-clients  [parámetro de libvirt-configuration]
Número máximo de conexiones concurrentes de clientes permitidas en todos los soc-
kets combinados.
El valor predeterminado es ‘5000’.

integer max-queued-clients  [parámetro de libvirt-configuration]
Longitud máxima de la cola de conexiones a la espera de ser aceptadas por el daemon.
Fíjese que algunos protocolos que implementan la retransmisión pueden obedecer esto
de manera que un intento posterior de conexión tenga éxito.
El valor predeterminado es ‘1000’.

integer max-anonymous-clients  [parámetro de libvirt-configuration]
Longitud máxima de la cola de clientes aceptados pero no identificados todavía. Pro-
porcione el valor cero para desactivar esta característica.
El valor predeterminado es ‘20’.
integer min-workers [parámetro de libvirt-configuration]
Número de procesos de trabajo que se lanzarán inicialmente.
El valor predeterminado es ‘5’.

integer max-workers [parámetro de libvirt-configuration]
Número máximo de hilos de trabajo.
Si el número de clientes excede min-workers, se lanzan más hilos, hasta el límite max-workers. Habitualmente se desea que max-workers sea igual al número máximo de clientes permitido.
El valor predeterminado es ‘20’.

integer prio-workers [parámetro de libvirt-configuration]
Número de procesos de trabajo prioritarios. Si todos los hilos de trabajo del conjunto previo se encuentran bloqueados, algunas llamadas marcadas como de alta prioridad (notablemente domainDestroy) pueden ejecutarse en este conjunto de hilos.
El valor predeterminado es ‘5’.

integer max-requests [parámetro de libvirt-configuration]
Límite global total de llamadas RPC concurrentes.
El valor predeterminado es ‘20’.

integer max-client-requests [parámetro de libvirt-configuration]
Límite de peticiones concurrentes desde una única conexión de cliente. Para evitar que un cliente monopolice el servidor esto debe ser una pequeña fracción de los parámetros globales “max_requests” y “max_workers”.
El valor predeterminado es ‘5’.

integer admin-min-workers [parámetro de libvirt-configuration]
Igual que min-workers pero para la interfaz de administración.
El valor predeterminado es ‘1’.

integer admin-max-workers [parámetro de libvirt-configuration]
Igual que max-workers pero para la interfaz de administración.
El valor predeterminado es ‘5’.

integer admin-max-clients [parámetro de libvirt-configuration]
Igual que max-clients pero para la interfaz de administración.
El valor predeterminado es ‘5’.

integer admin-max-queued-clients [parámetro de libvirt-configuration]
Igual que max-queued-clients pero para la interfaz de administración.
El valor predeterminado es ‘5’.

integer admin-max-client-requests [parámetro de libvirt-configuration]
Igual que max-client-requests pero para la interfaz de administración.
El valor predeterminado es ‘5’.
integer log-level  [parámetro de libvirt-configuration]
Nivel de registro. 4 errores, 3 avisos, 2 información, 1 depuración.
El valor predeterminado es ‘3’.

string log-filters  [parámetro de libvirt-configuration]
Filtros del registro.
Un filtro permite la selección de un nivel de registro diferente para una categoría dada de registros. El formato del filtro es uno de los siguientes:
• x:nombre
• x:+nombre

donde nombre es una cadena contra la que se compara la categoría proporcionada en la llamada VIR_LOG_INIT() al principio de cada fichero de fuentes de libvirt, por ejemplo "remote", "qemu" o "util.json" (el nombre en el filtro puede ser una subcadena del nombre completo de la categoría, para aceptar múltiples categorías con nombres similares), el prefijo opcional "+" indica a libvirt que registre la pila de llamadas en cada mensaje con el nombre correspondiente, y x es el nivel mínimo de los mensajes que deben registrarse:
• 1: DEBUG (depuración)
• 2: INFO (información)
• 3: WARNING (aviso)
• 4: ERROR

Se pueden definir en una única sentencia múltiples filtros, únicamente hace falta separarlos por espacios.
El valor predeterminado es ‘"3:remote 4:event"’.

string log-outputs  [parámetro de libvirt-configuration]
Salidas de log.
Una salida es uno de esos lugares para almacenar información de logging. El formato para una salida puede ser:
x:stderr  la salida va a stderr
x:syslog:nombre  usa syslog para la salida y usa el nombre proporcionado como identificador
x:file:ruta_fichero  encamina la salida a un fichero, con la ruta proporcionada
x:journald  usa el sistema de logging journald

En todos los casos el prefijo x es el nivel mínimo, que actúa como filtro
• 1: DEBUG (depuración)
• 2: INFO (información)
• 3: WARNING (aviso)
• 4: ERROR
Se pueden definir salidas múltiples, únicamente deben separarse por espacios.
El valor predeterminado es "3:stderr".

**integer audit-level** [parámetro de libvirt-configuration]
Permite la alteración del uso del sistema de auditoría.
- 0: desactiva la auditoría
- 1: activa la auditoría, únicamente si está activado en la máquina
- 2: activa la auditoría, y sale si está desactivada en la máquina.
El valor predeterminado es ‘1’.

**boolean audit-logging** [parámetro de libvirt-configuration]
Envía los mensajes de auditoría a través de la infraestructura de registro de libvirt.
El valor predeterminado es ‘#f’

**string-opcional host-uuid** [parámetro de libvirt-configuration]
 UUID de la máquina anfitriona. No pueden ser iguales todos los dígitos del UUID.
El valor predeterminado es ‘”’.

**string host-uuid-source** [parámetro de libvirt-configuration]
Fuente de lectura del UUID de la máquina anfitriona.
- **smbios**: obtiene el UUID de **dmidecode -s system-uuid**
- **machine-id**: obtiene el UUID de **/etc/machine-id**
Si **dmidecode** no proporciona un UUID válido, se generará un UUID temporal.
El valor predeterminado es ‘"smbios"’.

**integer keepalive-interval** [parámetro de libvirt-configuration]
Un mensaje “keepalive” se envía al cliente tras **keepalive_interval** segundos de inactividad para comprobar si el cliente todavía responde. Si se proporciona el valor -1, libvirtd nunca enviará peticiones “keepalive”; no obstante los clientes todavía pueden mandarlas y el daemon enviará las respuestas.
El valor predeterminado es ‘5’.

**integer keepalive-count** [parámetro de libvirt-configuration]
Número máximo de mensajes “keepalive” que se permite enviar a un cliente sin obtener respuesta antes de considerar que se ha roto la conexión.
En otras palabras, la conexión se cierra automáticamente tras **keepalive_interval** * (keepalive_count + 1) segundos tras la última recepción de un mensaje desde el cliente. Cuando **keepalive_count** tiene valor 0, las conexiones se cerrarán automáticamente tras **keepalive-interval** segundos de inactividad sin mandar ningún mensaje “keepalive”.
El valor predeterminado es ‘5’.

**integer admin-keepalive-interval** [parámetro de libvirt-configuration]
Igual que la opción anterior pero para la interfaz de administración.
El valor predeterminado es ‘5’.
integer admin-keepalive-count  [parámetro de libvirt-configuration]
Igual que la opción anterior pero para la interfaz de administración.
El valor predeterminado es ‘5’.

integer ovs-timeout  [parámetro de libvirt-configuration]
Plazo máximo para las llamadas a Open vSwitch.
La utilidad ovs-vsctl se usa para la configuración y su opción de plazo máximo (timeout) tiene un valor de 5 segundos de manera predeterminada para evitar que esperas potencialmente infinitas bloquee libvirt.
El valor predeterminado es ‘5’.

Daemon Virtlog
El servicio virtlogd es un daemon del que se compone el lado servidor de libvirt cuya finalidad es la gestión del registro de las consolas de las máquinas virtuales.
Este daemon no se usa directamente en aplicaciones cliente de libvirt, sino que libvirtd lo invoca en su nombre. Al mantener los registros en un daemon independiente, el daemon libvirtd puede reiniciarse sin correr el riesgo de perder registros. El daemon virtlogd tiene la capacidad de ejecutar exec() sobre sí mismo al recibir SIGUSR1, para permitir actualizaciones en vivo sin tiempo de parada.

virtlog-service-type  [Variable Scheme]
Este es el tipo del daemon virtlog. Su valor debe ser un objeto virtlog-configuration.

   (service virtlog-service-type
     (virtlog-configuration
      (max-clients 1000)))

integer log-level  [parámetro de virtlog-configuration]
Nivel de registro. 4 errores, 3 avisos, 2 información, 1 depuración.
El valor predeterminado es ‘3’.

string log-filters  [parámetro de virtlog-configuration]
Filtros del registro.
Un filtro permite la selección de un nivel de registro diferente para una categoría dada de registros. El formato del filtro es uno de los siguientes:

- x:nombre
- x:+nombre

donde nombre es una cadena contra la que se compara la categoría proporcionada en la llamada VIR_LOG_INIT() al principio de cada fichero de fuentes de libvirt, por ejemplo "remote", "qemu" o "util.json" (el nombre en el filtro puede ser una subcadena del nombre completo de la categoría, para aceptar múltiples categorías con nombres similares), el prefijo opcional "+" indica a libvirt que registre la pila de llamadas en cada mensaje con el nombre correspondiente, y x es el nivel mínimo de los mensajes que deben registrarse:

- 1: DEBUG (depuración)
• 2: INFO (información)
• 3: WARNING (aviso)
• 4: ERROR

Se pueden definir en una única sentencia múltiples filtros, únicamente hace falta separarlos por espacios.

El valor predeterminado es ‘"3:remote 4:event"’.

**string log-outputs**  
[parámetro de virtlog-configuration]

Salidas de log.

Una salida es uno de esos lugares para almacenar información de logging. El formato para una salida puede ser:

x:stderr la salida va a stderr

x:syslog:nombre  
usa syslog para la salida y usa el nombre proporcionado como identificador

x:file:ruta_fichero  
encamina la salida a un fichero, con la ruta proporcionada

x:journald  
usa el sistema de logging journald

En todos los casos el prefijo x es el nivel mínimo, que actúa como filtro

• 1: DEBUG (depuración)
• 2: INFO (información)
• 3: WARNING (aviso)
• 4: ERROR

Se pueden definir salidas múltiples, únicamente deben separarse por espacios.

El valor predeterminado es ‘"3:stderr"’.

**integer max-clients**  
[parámetro de virtlog-configuration]

Número máximo de conexiones concurrentes de clientes permitidas en todos los sockets combinados.

El valor predeterminado es ‘1024’.

**integer max-size**  
[parámetro de virtlog-configuration]

Tamaño máximo del fichero antes de pasar al siguiente.

El valor predeterminado es ‘2MB’.

**integer max-backups**  
[parámetro de virtlog-configuration]

Número máximo de ficheros de backup que se deben mantener.

El valor predeterminado es ‘3’.
Emulación transparente con QEMU

qemu-binfmt-service-type proporciona la capacidad de emular transparentemente programas binarios construidos para arquitecturas diferentes—por ejemplo, le permite ejecutar de manera transparente un programa de ARMv7 en una máquina x86_64. Ésto se consigue mediante la combinación del emulador QEMU (https://www.qemu.org) y la característica binfmt_misc del núcleo Linux.

qemu-binfmt-service-type [Variable Scheme]
Este es el tipo del servicio de emulación transparente QEMU/binfmt. Su valor debe ser un objeto qemu-binfmt-configuration, que especifica el paquete QEMU usado así como las arquitecturas que se desean emular:

\[
\text{(service qemu-binfmt-service-type} \\
\text{ (qemu-binfmt-configuration} \\
\text{ (platforms (lookup-qemu-platforms "arm" "aarch64" "mips64el")))})\]

En este ejemplo se activa la emulación transparente para las plataformas ARM y aarch64. La ejecución de herd stop qemu-binfmt la desactiva, y la ejecución de herd start qemu-binfmt la vuelve a activar (véase Sección “Invoking herd” en The GNU Shepherd Manual).

qemu-binfmt-configuration [Tipo de datos]
Esta es la configuración para el servicio qemu-binfmt.

platforms (predeterminadas: ') Lista de plataformas de QEMU emuladas. Cada elemento debe ser un objeto de plataforma como los devueltos por lookup-qemu-platforms (véase a continuación).

guix-support? (predeterminado: #f)
Cuando su valor es verdadero, QEMU y todas sus dependencias se añaden al entorno de construcción de guix-daemon (véase Sección 2.5 [Invocación de guix-daemon], página 15). Esto permite que se usen los controladores binfmt_misc dentro del entorno de construcción, lo que permite construir transparentemente programas para otra arquitectura.

Por ejemplo, supongamos que está en una máquina x86_64 y tiene este servicio:

\[
\text{(service qemu-binfmt-service-type} \\
\text{ (qemu-binfmt-configuration} \\
\text{ (platforms (lookup-qemu-platforms "arm"))} \\
\text{ (guix-support? #t))})\]

Puede ejecutar:

guix build -s armhf-linux inkscape
Y construirá Inkscape para ARMv7 como si fuese una construcción nativa, de manera transparente mediante el uso de QEMU para emular la CPU ARMv7. ¡Muy útil si quisiera probar la construcción de un paquete en una arquitectura a la que no tenga acceso!

qemu (predeterminado: qemu)
El paquete QEMU usado.
lookup-qemu-platforms plataformas...  
[Procedimiento Scheme]  
Devuelve la lista de objetos de plataforma de QEMU que corresponden a plataformas... plataformas debe ser una lista de cadenas que correspondan con nombres de plataforma, como "arm", "sparc", "mips64el", etcétera.

qemu-platform? obj  
[Procedimiento Scheme]  
Devuelve verdadero si obj es un objeto de plataforma.

qemu-platform-name plataforma  
[Procedimiento Scheme]  
Devuelve el nombre de plataforma—una cadena como "arm".

8.8.25 Servicios de control de versiones

El módulo (gnu services version-control) proporciona un servicio para permitir el acceso remoto a repositorios Git locales. Existen tres opciones: el servicio git-daemon-service, que proporciona acceso a repositorios a través del protocolo inseguro basado en TCP git://, la extensión del servidor web nginx para redirigir algunas peticiones al motor git-http-backend, o una interfaz web proporcionada por el servicio cgit-service-type.

git-daemon-service [#:config  
(git-daemon-configuration)]  
[Procedimiento Scheme]  
Devuelve un servicio que ejecuta git daemon, un servidor TCP simple para exponer repositorios con el protocolo Git para acceso anónimo.

El parámetro opcional config debe ser un objeto <git-daemon-configuration>, de manera predeterminada permite acceso de solo lectura a los repositorios exportados bajo /srv/git.

git-daemon-configuration  
[Tipo de datos]  
Tipo de datos que representa la configuración para git-daemon-service.

package (predeterminado: git)  
El objeto paquete del sistema distribuido de control de versiones Git.

export-all? (predeterminado: #f)  
Determina si se permite el acceso a todos los repositorios Git, incluso si no tienen el fichero git-daemon-export-ok.

base-path (predeterminado: /srv/git)  
Determina si se traducirán todas las rutas de las peticiones como relativas a la ruta proporcionada. Si se encuentra en ejecución el daemon de git con (base-path " /srv/git" en example.com, al realizar la solicitud de git://example.com/hello.git, el daemon de git interpretará la ruta como /srv/git/hello.git.

user-path (predeterminado: #f)  
Determina si se permite el uso de la notación ~user en las peticiones. Si se especifica una cadena vacía, una petición de git://máquina/~alicia/algo se tomará como una petición de acceso al repositorio algo en el directorio de la usuaria alicia. Si se especifica

17 Mediante la creación del fichero mágico "git-daemon-export-ok" en el directorio del repositorio.
Capítulo 8: Configuración del sistema

(\textit{user-path "ruta"}), la misma petición se traducirá en una petición de acceso al repositorio \textit{ruta/algo} en el directorio de la usuaria \textit{alicia}.

**listen** (predeterminado: '{})
Determina si se debe escuchar en direcciones IP o nombres de máquina específicos, de manera predeterminada escucha en cualquiera.

**port** (predeterminado: '#'f)
Determina si se escucha en un puerto alternativo, cuyo valor predeterminado es 9418.

**whitelist** (predeterminado: '{})
Si no está vacío, únicamente permite el acceso a esta lista de directorios.

**extra-options** (predeterminadas: '{})
Opciones adicionales que se proporcionan a \texttt{git daemon}, para obtener más información le rogamos que ejecute \texttt{man git-daemon}.

El protocolo \texttt{git://} carece de verificación. Cuando se obtienen datos de un repositorio a través del protocolo \texttt{git://}, no puede tener plena confianza en que los datos que reciba procedan realmente de la máquina que ha indicado, y su conexión puede estar sujeta a interceptaciones. Es mejor usar un transporte verificado y cifrado, como \texttt{https}. Aunque Git le permite servir repositorios usando servidores web poco sofisticados basados en ficheros, existe un protocolo más rápido implementado en el programa \texttt{git-http-backend}. Este programa es el motor de un servicio web de Git adecuado. Está diseñado para ejecutarse tras FastCGI. Véase Sección 8.8.16 [Servicios Web], página 303, para más información sobre la ejecución del daemon \texttt{fcgiwrap} necesario.

Guix tiene un tipo de datos de configuración distinto para proporcionar repositorios Git sobre HTTP.

**git-http-configuration**

[Tipo de datos]

Tipo de datos que representa la configuración para \texttt{git-http-service}.

**package** (predeterminado: \texttt{git})
El objeto paquete del sistema distribuido de control de versiones Git.

**git-root** (predeterminada: /srv/git)
Directorio que contiene los repositorios Git que se expondrán al mundo.

**export-all?** (predeterminado: '#'f)
Determina si se expondrá el acceso a todos los repositorios en \texttt{git-root}, incluso si no contienen el fichero \texttt{git-daemon-export-ok}.

**uri-path** (predeterminada: /git/)
Prefijo de la ruta del acceso de Git. Con el prefijo predeterminado /\texttt{git/}, \url{http://servidor/git/repositorio.git} se traducirá en /srv/git/repositorio.git. Las peticiones cuyas rutas URI no comiencen con dicho prefijo no se pasan a esta instancia de Git.

**fcgiwrap-socket** (predeterminado: 127.0.0.1:9000)
Socket en el que el daemon \texttt{fcgiwrap} escucha. Véase Sección 8.8.16 [Servicios Web], página 303.
No existe actualmente `git-http-service-type`; en vez de eso puede crear una configuración `nginx-location-configuration` desde `git-http-configuration` y añadir dicha configuración al servidor web.

**Procedimiento Scheme**

\[
\text{git-http-nginx-location-configuration} \quad \text{[Procedimiento Scheme]}
\]

Calcula una configuración `nginx-location-configuration` que corresponde con la configuración http de Git proporcionada. Un ejemplo de definición de servicio nginx que ofrece el directorio predeterminado `/srv/git` sobre HTTPS podría ser:

\[
\text{(service nginx-service-type)}
\]

\[
\begin{align*}
\text{(server-configuration)}
\end{align*}
\]

\[
\begin{align*}
\text{(nginx-server-configuration)}
\end{align*}
\]

\[
\begin{align*}
\text{(listen }'("443 \text{ ssl}))
\end{align*}
\]

\[
\begin{align*}
\text{(server-name }"\text{git.mi-maquina.org}"
\end{align*}
\]

\[
\begin{align*}
\text{(ssl-certificate}
\end{align*}
\]

\[
\begin{align*}
"/\text{etc/letsencrypt/live/git.mi-maquina.org/fullchain.pem})
\end{align*}
\]

\[
\begin{align*}
\text{(ssl-certificate-key}
\end{align*}
\]

\[
\begin{align*}
"/\text{etc/letsencrypt/live/git.mi-maquina.org/privkey.pem})
\end{align*}
\]

\[
\begin{align*}
\text{(locations}
\end{align*}
\]

\[
\begin{align*}
\text{(list}
\end{align*}
\]

\[
\begin{align*}
\text{(git-http-nginx-location-configuration}
\end{align*}
\]

\[
\begin{align*}
\text{(git-http-configuration (uri-path "/"))})
\end{align*}
\]

Este ejemplo asume que usa Let’s Encrypt para obtener su certificado TLS. Véase Sección 8.8.17 [Servicios de certificados], página 320. El servicio predeterminado `certbot` redirigirá todo el tráfico HTTP en `git.mi-maquina.org` hacia HTTPS. También necesitará añadir `fcgiwrap` a sus servicios del sistema. Véase Sección 8.8.16 [Servicios Web], página 303.

**Servicio Cgit**

Cgit ([https://git.zx2c4.com/cgit/](https://git.zx2c4.com/cgit/)) es un servidor de fachada para repositorios Git escrito en C.

El ejemplo siguiente configura el servicio con los valores predeterminados. Por omisión, se puede acceder a Cgit en el puerto 80 (`http://localhost:80`).

\[
\text{(service cgit-service-type)}
\]

El tipo `file-object` designa o bien un objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108), o bien una cadena.

Los campos disponibles de `cgit-configuration` son:

- `package package` [parámetro de `cgit-configuration`]
  El paquete CGIT.

- `lista-nginx-server-configuration` [parámetro de `cgit-configuration`]
  `nginx`
  Configuración de NGINX.
file-object about-filter [parámetro de cgit-configuration]
Especifica una orden que se llamará para dar formato al contenido de las páginas “about” (tanto al nivel superior como cada repositorio).
El valor predeterminado es ‘”’.

string agefile [parámetro de cgit-configuration]
Especifica una ruta, relativa a cada ruta de repositorio, que puede usarse para especificar una fecha y hora de la revisión más reciente del repositorio.
El valor predeterminado es ‘”’.

file-object auth-filter [parámetro de cgit-configuration]
Especifica una orden que se invocará para la validación de acceso al repositorio.
El valor predeterminado es ‘”’.

string branch-sort [parámetro de cgit-configuration]
Opción que, cuando tiene valor ‘age’, activa la ordenación por fecha en la lista de referencias de ramas, y cuando tiene valor ‘name’ activa la ordenación por nombre de rama.
El valor predeterminado es ‘”name”’.

string cache-root [parámetro de cgit-configuration]
Ruta usada para el almacenamiento de las entradas de caché de cgit.
El valor predeterminado es ‘”/var/cache/cgit”’.

integer cache-static-ttl [parámetro de cgit-configuration]
Número que especifica el tiempo de vida, en minutos, de la versión en caché de las páginas del repositorio accedidas mediante un hash SHA1 fijo.
El valor predeterminado es ‘’-’1’’.

integer cache-dynamic-ttl [parámetro de cgit-configuration]
Número que especifica el tiempo de vida, en minutos, de la versión en caché de las páginas del repositorio accedidas sin un hash SHA1 fijo.
El valor predeterminado es ‘’5’’.

integer cache-repo-ttl [parámetro de cgit-configuration]
Número que especifica el tiempo de vida, en minutos, de la versión en caché de la página de resumen del repositorio.
El valor predeterminado es ‘’5’’.

integer cache-root-ttl [parámetro de cgit-configuration]
Número que especifica el tiempo de vida, en minutos, de la versión en caché de la página del índice de repositorios.
El valor predeterminado es ‘’5’’.

integer cache-scanrc-ttl [parámetro de cgit-configuration]
Número que especifica el tiempo de vida, en minutos, para el resultado de la búsqueda en una ruta para repositorios Git.
El valor predeterminado es ‘’15’’. 
integer cache-about-ttl  [parámetro de cgit-configuration]  
Número que especifica el tiempo de vida, en minutos, de la versión en caché de la página de información del repositorio.
El valor predeterminado es ‘15’.

integer cache-snapshot-ttl  [parámetro de cgit-configuration]  
Número que especifica el tiempo de vida, en minutos, de la versión en caché de las instantáneas.
El valor predeterminado es ‘5’.

integer cache-size  [parámetro de cgit-configuration]  
El número máximo de entradas en la caché de cgit. Cuando el valor es ‘0’, se desactiva el almacenamiento en caché.
El valor predeterminado es ‘0’.

boolean case-sensitive-sort?  [parámetro de cgit-configuration]  
Ordena los elementos en la lista del repositorio diferenciando las mayúsculas.
El valor predeterminado es ‘#t’

lista clone-prefix  [parámetro de cgit-configuration]  
Lista de prefijos comunes que, cuando se combinan con la URL de un repositorio, generan una URL que permite el clonado del repositorio.
El valor predeterminado es ‘()’.

lista clone-url  [parámetro de cgit-configuration]  
Lista de plantillas clone-url.
El valor predeterminado es ‘()’.

file-object commit-filter  [parámetro de cgit-configuration]  
Orden ejecutada para el formato de mensajes de revisión.
El valor predeterminado es ‘”’.

string commit-sort  [parámetro de cgit-configuration]  
Opción que, cuando tiene valor ‘date’, activa la ordenación estricta por fecha en el registro histórico de revisiones, y cuando tiene valor ‘topo’ activa la ordenación estricta topológica.
El valor predeterminado es ‘”git log”’.

file-object css  [parámetro de cgit-configuration]  
URL que especifica el documento css incluido en todas las páginas de cgit.
El valor predeterminado es ‘”/share/cgit/cgit.css”’.

file-object email-filter  [parámetro de cgit-configuration]  
Especifica una orden que se llamará para dar formato a los nombres y las direcciones de correo electrónico de las revisoras, autoras y etiquetadoras con el que se representarán en varios lugares de la interfaz cgit.
El valor predeterminado es ‘”’.
boolean embedded? [parámetro de cgit-configuration]
Opción que, cuando tiene valor '#t', hace que cgit genere un fragmento HTML adecuado para embeberse en otras páginas HTML.
El valor predeterminado es '#f'

boolean enable-commit-graph? [parámetro de cgit-configuration]
Opción que, cuando tiene el valor '#t', hace que cgit imprima un grafo histórico de la revisión de arte ASCII a la izquierda de los mensajes de revisión en la página del histórico del repositorio.
El valor predeterminado es '#f'

boolean enable-filter-overrides? [parámetro de cgit-configuration]
Opción que, cuando tiene valor '#t', permite que todas las configuraciones de filtros se sustituyan en los ficheros cgitrc específicos del repositorio.
El valor predeterminado es '#f'

boolean enable-follow-links? [parámetro de cgit-configuration]
Opción que, cuando tiene valor '#t', permite a las usuarias seguir un fichero en la vista de registro (log).
El valor predeterminado es '#f'

boolean enable-http-clone? [parámetro de cgit-configuration]
Si se proporciona '#t', cgit actuará como un simple servidor HTTP para los clones de Git.
El valor predeterminado es '#t'

boolean enable-index-links? [parámetro de cgit-configuration]
Opción que, cuando tiene valor '#t', hace que cgit genere enlaces adicionales "summary" (resumen), "commit" (revisión) y "tree" (árbol) para cada repositorio en el índice de repositorios.
El valor predeterminado es '#f'

boolean enable-index-owner? [parámetro de cgit-configuration]
Opción que, cuando tiene valor '#t', hace que cgit muestre la propietaria de cada repositorio en el índice del repositorios.
El valor predeterminado es '#t'

boolean enable-log-filecount? [parámetro de cgit-configuration]
Opción que, cuando se proporciona el valor '#t', hace que cgit imprima el número de ficheros modificados por cada revisión en la página de registro histórico del repositorio ("log").
El valor predeterminado es '#f'

boolean enable-log-linecount? [parámetro de cgit-configuration]
Opción que, cuando se proporciona el valor '#t', hace que cgit imprima el número de líneas añadidas y eliminadas en cada revisión en la página de registro histórico ("log").
El valor predeterminado es '#f'
Capítulo 8: Configuración del sistema

boolean enable-remote-branches? [parámetro de cgit-configuration]
Opción que, cuando se proporciona el valor ‘#t’, hace que cgit muestre ramas remotas
en las vistas de resumen ("summary") y de referencias ("refs").
El valor predeterminado es ‘#f’

boolean enable-subject-links? [parámetro de cgit-configuration]
Opción que, cuando se proporciona el valor ‘1’, hace que cgit use el asunto de la
revisión previa como texto del enlace cuando se generen enlaces a revisiones previas
en la vista de la revisión.
El valor predeterminado es ‘#f’

boolean enable-html-serving? [parámetro de cgit-configuration]
Opción que, cuando se proporciona el valor ‘#t’, hace que cgit use el asunto de la
revisión previa como texto del enlace cuando se generen enlaces a revisiones previas
en la vista de la revisión.
El valor predeterminado es ‘#f’

boolean enable-tree-linenumbers? [parámetro de cgit-configuration]
Opción que, cuando se proporciona el valor ‘#t’, hace que cgit genere enlaces de
números de línea para los ficheros (blob) de texto plano impresos en la vista de árbol.
El valor predeterminado es ‘#t’

boolean enable-git-config? [parámetro de cgit-configuration]
Opción que, cuando tiene valor ‘#f’, permite que cgit use la configuración de Git para
fijar el valor de cualquier opción específica del repositorio.
El valor predeterminado es ‘#f’

file-object favicon [parámetro de cgit-configuration]
URL usada para icono de los enlaces a cgit.
El valor predeterminado es ‘"/favicon.ico"’.

string footer [parámetro de cgit-configuration]
El contenido del fichero especificado con esta opción se incluirá literalmente en la
parte inferior de todas las páginas (es decir, sustituye al mensaje estándar "generated by...").
El valor predeterminado es ‘""’.

string head-include [parámetro de cgit-configuration]
El contenido del fichero especificado con esta opción se incluirá literalmente en la
sección HEAD de HTML en todas las páginas.
El valor predeterminado es ‘""’.

string header [parámetro de cgit-configuration]
El contenido del fichero especificado con esta opción se incluirá literalmente en la
parte superior de todas las páginas.
El valor predeterminado es ‘""’.
Capítulo 8: Configuración del sistema

file-object include [parámetro de cgit-configuration]
Nombre de un fichero de configuración que debe incluirse antes de procesar el resto del fichero de configuración actual.
El valor predeterminado es ‘”’.

string index-header [parámetro de cgit-configuration]
El contenido del fichero especificado en esta opción se incluirá literalmente sobre el índice de repositorios.
El valor predeterminado es ‘”’.

string index-info [parámetro de cgit-configuration]
El contenido del fichero especificado con esta opción se incluirá de manera literal bajo la cabecera en la página de índice del repositorio.
El valor predeterminado es ‘”’.

boolean local-time? [parámetro de cgit-configuration]
Opción que, cuando tiene valor ‘#t’, hace que cgit imprima las fechas de revisión y etiqueta en la zona horaria del servidor.
El valor predeterminado es ‘#f’

file-object logo [parámetro de cgit-configuration]
URL que especifica la fuente de una imagen usada como logo en todas las páginas de cgit.
El valor predeterminado es ‘”/share/cgit/cgit.png”’.

string logo-link [parámetro de cgit-configuration]
URL que se carga al pulsar la imagen del logo de cgit.
El valor predeterminado es ‘”’.

file-object owner-filter [parámetro de cgit-configuration]
Orden que se ejecuta para dar formato a la columna de propietaria (Owner) de la página principal.
El valor predeterminado es ‘”’.

integer max-atom-items [parámetro de cgit-configuration]
Número de elementos a mostrar en la vista de “atom feeds”.
El valor predeterminado es ‘10’.

integer max-commit-count [parámetro de cgit-configuration]
Número de entradas a mostrar por página en la vista del registro histórico ("log").
El valor predeterminado es ‘50’.

integer max-message-length [parámetro de cgit-configuration]
Número de caracteres del mensaje de la revisión a mostrar en la vista del registro histórico ("log").
El valor predeterminado es ‘80’.
integer max-repo-count [parámetro de cgit-configuration]
Especifica el número de entradas a mostrar por página en la página de índice de repositorios.
El valor predeterminado es ‘50’.

integer max-repodesc-length [parámetro de cgit-configuration]
Especifica el número máximo de caracteres mostrados en la descripción del repositorio en la página del índice de repositorios.
El valor predeterminado es ‘80’.

integer max-blob-size [parámetro de cgit-configuration]
Especifica el tamaño máximo de un fichero (blob) para mostrarlo en HTML en kilobytes.
El valor predeterminado es ‘0’.

string max-stats [parámetro de cgit-configuration]
El valor predeterminado es ‘”’.

mimetype-alist mimetype [parámetro de cgit-configuration]
Tipo MIME para la extensión de fichero especificada.
El valor predeterminado es ‘((gif "image/gif") (html "text/html")
(jpg "image/jpeg") (jpeg "image/jpeg") (pdf "application/pdf") (png
"image/png") (svg "image/svg+xml”))’.

file-object mimetype-file [parámetro de cgit-configuration]
Especifica el fichero usado para la búsqueda automática de tipos MIME.
El valor predeterminado es ‘”’.

string module-link [parámetro de cgit-configuration]
Texto que se usará como la cadena de formato para un enlace cuando un submódulo se imprime en el listado del directorio.
El valor predeterminado es ‘”’.

boolean nocache? [parámetro de cgit-configuration]
Si se proporciona el valor ‘#t’, se desactiva la caché.
El valor predeterminado es ‘#f’

boolean noplainemail? [parámetro de cgit-configuration]
Si se proporciona ‘#t’, se desactiva la impresión de direcciones de correo completas de las autoras.
El valor predeterminado es ‘#f’

boolean noheader? [parámetro de cgit-configuration]
Opción que, cuando tiene valor ‘#t’, hace que cgit omita la cabecera estándar en todas las páginas.
El valor predeterminado es ‘#f’
lista-proyectos project-list [parámetro de cgit-configuration]
Una lista de subdirectorios dentro de repository-directory, relativa a él, que debe cargarse como repositorios Git. La lista vacía significa que se cargarán todos los subdirectorios.
El valor predeterminado es ‘()’.

file-object readme [parámetro de cgit-configuration]
Texto usado como valor predeterminado para cgit-repo-readme.
El valor predeterminado es ‘”’.

boolean remove-suffix?
Si se proporciona #t y repository-directory está activo, si se encuentra algún repositorio con el sufijo .git, se elimina dicho sufijo de la URL y del nombre.
El valor predeterminado es ‘#f’.

integer renamelimit [parámetro de cgit-configuration]
Número máximo de ficheros considerados durante la detección de renombrados.
El valor predeterminado es ‘-1’.

string repository-sort [parámetro de cgit-configuration]
La forma de ordenar los repositorios de cada sección.
El valor predeterminado es ‘”’.

lista-robots robots [parámetro de cgit-configuration]
Texto usado como contenido de la meta-etiqueta robots.
El valor predeterminado es ‘(”noindex" "nofollow”)’.

string root-desc [parámetro de cgit-configuration]
Texto impreso bajo la cabecera en la página de índice del repositorio.
El valor predeterminado es ‘"a fast webinterface for the git dscm"’.

string root-readme [parámetro de cgit-configuration]
The content of the file specified with this option will be included verbatim below the "about" link on the repository index page.
El valor predeterminado es ‘"”’.

string root-title [parámetro de cgit-configuration]
Texto impreso como cabecera en la página de índice del repositorio.
El valor predeterminado es ‘”’.

boolean scan-hidden-path [parámetro de cgit-configuration]
Si se proporciona ‘#t’ y repository-directory está activo, repository-directory recorrerá recursivamente los directorios cuyos nombres comiencen por punto. En otro caso, repository-directory no tendrá en cuenta dichos directorios, considerados “ocultos”.
Tenga en cuenta que esto no incluye al directorio “.git” en repositorios con una copia de trabajo.
El valor predeterminado es ‘#f’.
lista snapshots [parámetro de cgit-configuration]
Texto que especifica el conjunto predeterminado de formatos de instantánea para los que cgit genera enlaces.
El valor predeterminado es ‘()’.

directorio-repositorio [parámetro de cgit-configuration]
repository-directory
Nombre del directorio en el que se buscarán repositorios (representa scan-path).
El valor predeterminado es ‘"/srv/git"’.

string section [parámetro de cgit-configuration]
Nombre actual de la sección de repositorios - todos los repositorios definidos tras esta opción heredarán el nombre actual de sección.
El valor predeterminado es ‘""’.

string section-sort [parámetro de cgit-configuration]
Opción que, cuando tiene valor ‘1’, ordenará las secciones en el listado de repositorios por nombre.
El valor predeterminado es ‘""’.

integer section-from-path [parámetro de cgit-configuration]
Número que, si se define antes de repository-directory, especifica cuantos elementos de ruta de cada ruta de repositorio se usarán como nombre de sección predeterminado.
El valor predeterminado es ‘0’.

boolean side-by-side-diffs? [parámetro de cgit-configuration]
Si se proporciona el valor ‘#t’ se muestran las diferencias lado a lado en vez de usar el formato universal de manera predeterminada.
El valor predeterminado es ‘#f’

file-object source-filter [parámetro de cgit-configuration]
Especifica la orden que se ejecutará para dar formato a los ficheros (blob) de texto plano en la vista de árbol.
El valor predeterminado es ‘""’.

integer summary-branches [parámetro de cgit-configuration]
Especifica el número de ramas mostradas en la vista resumen ("summary") del repositorio.
El valor predeterminado es ‘10’.

integer summary-log [parámetro de cgit-configuration]
Especifica el número de entradas del registro mostradas en la vista resumen ("summary") del repositorio.
El valor predeterminado es ‘10’.

integer summary-tags [parámetro de cgit-configuration]
Especifica el número que etiquetas que se mostrarán en la vista resumen ("summary") del repositorio.
El valor predeterminado es ‘10’.
Capítulo 8: Configuración del sistema

string strict-export [parámetro de cgit-configuration]
Nombre de fichero que, en caso de especificarse, debe estar presente en el repositorio para que se permita el acceso de cgit a dicho repositorio.
El valor predeterminado es ‘”’.

string virtual-root [parámetro de cgit-configuration]
URL que, en caso de especificarse, se usará como raíz de todos los enlaces de cgit.
El valor predeterminado es ‘”/”’.

lista-repository-cgit-configuration [parámetro de cgit-configuration]
repositories
Lista de registros cgit-repo usados con la configuración.
El valor predeterminado es ‘()’.
Los campos disponibles de repository-cgit-configuration son:

repo-list snapshots [parámetro de repository-cgit-configuration]
Una máscara de los formatos de instantánea para este repositorio para los que cgit genera enlaces, restringida por la opción de configuración global snapshots.
El valor predeterminado es ‘()’.

repo-file-object source-filter [parámetro de repository-cgit-configuration]
Sustituye al valor predeterminado de source-filter.
El valor predeterminado es ‘”’.

repo-string url [parámetro de repository-cgit-configuration]
La URL relativa usada para el acceso al repositorio.
El valor predeterminado es ‘”’.

repo-file-object about-filter [parámetro de repository-cgit-configuration]
Sustituye al valor predeterminado de about-filter.
El valor predeterminado es ‘”’.

repo-string branch-sort [parámetro de repository-cgit-configuration]
Cuando se proporciona el valor ‘age’, activa la ordenación por fecha en la lista de referencias de ramas, y cuando se proporciona ‘name’ se activa la ordenación por nombre de rama.
El valor predeterminado es ‘”’.

repo-list clone-url [parámetro de repository-cgit-configuration]
Una lista de URL que se pueden usar para clonar el repositorio.
El valor predeterminado es ‘()’.

repo-file-object commit-filter [parámetro de repository-cgit-configuration]
Sustituye al valor predeterminado de commit-filter.
El valor predeterminado es ‘”’.
Capítulo 8: Configuración del sistema 376

repo-string commit-sort [parámetro de repository-cgit-configuration]
Opción que, cuando tiene valor ‘date’, activa la ordenación estricta por fecha en el registro histórico de revisiones, y cuando tiene valor ‘topo’ activa la ordenación estricta topológica.
El valor predeterminado es ‘”’.

repo-string defbranch [parámetro de repository-cgit-configuration]
Nombre de la rama predeterminada de este repositorio. Si no existe dicha rama en el repositorio, se usará como predeterminado el primer nombre de rama encontrado (tras su ordenación). De manera predeterminada, la rama a la que apunta HEAD, o "master" si no existe un valor adecuado para HEAD.
El valor predeterminado es ‘”’.

repo-string desc [parámetro de repository-cgit-configuration]
El valor a mostrar como descripción del repositorio.
El valor predeterminado es ‘”’.

repo-string homepage [parámetro de repository-cgit-configuration]
El valor a mostrar como página web principal del repositorio.
El valor predeterminado es ‘”’.

repo-file-object email-filter
email-filter
Sustituye al valor predeterminado de email-filter.
El valor predeterminado es ‘”’.

maybe-repo-boolean enable-commit-graph?
Esta opción se puede usar para forzar el valor de la opción de configuración global enable-commit-graph?.
El valor predeterminado es ‘disabled’.

maybe-repo-boolean enable-log-filecount?
Esta opción se puede usar para forzar el valor de la opción de configuración global enable-log-filecount?.
El valor predeterminado es ‘disabled’.

maybe-repo-boolean enable-log-linecount?
Esta opción se puede usar para forzar el valor de la opción de configuración global enable-log-linecount?.
El valor predeterminado es ‘disabled’.

maybe-repo-boolean enable-remote-branches?
Opción que, cuando se proporcione el valor ‘#t’, hace que cgit muestre ramas remotas en las vistas de resumen ("summary") y de referencias ("refs")
El valor predeterminado es ‘disabled’.
maybe-repo-boolean enable-subject-links?
Esta opción se puede usar para forzar el valor de la opción de configuración global enable-subject-links?.
El valor predeterminado es ‘disabled’.

maybe-repo-boolean enable-html-serving?
Esta opción se puede usar para forzar el valor de la opción de configuración global enable-html-serving?.
El valor predeterminado es ‘disabled’.

repo-boolean hide?
Opción que, cuando tiene valor ‘#t’, oculta el repositorio en el índice.
El valor predeterminado es ‘#f’

repo-boolean ignore?
Opción que, cuando tiene valor ‘#t’, ignora el repositorio.
El valor predeterminado es ‘#f’

repo-file-object logo
URL que especifica la fuente de una imagen que se usará como logo en las páginas de este repositorio.
El valor predeterminado es ‘”’.

repo-string logo-link
URL que se carga al pulsar la imagen del logo de cgit.
El valor predeterminado es ‘”’.

repo-file-object owner-filter
Sustituye al valor predeterminado de owner-filter.
El valor predeterminado es ‘”’.

repo-string module-link
Texto que se usará como la cadena de formato de un enlace cuando un submódulo se imprima en el listado de un directorio. Los parámetros para la cadena de formato son la ruta y el SHA1 de la revisión del submódulo.
El valor predeterminado es ‘”’.

ruta-enlace-módulo module-link
Texto que se usará como la cadena de formato de un enlace cuando un submódulo con la ruta de subdirectorio especificada se imprima en el listado de un directorio.
El valor predeterminado es ‘()’.
repo-string max-stats [parámetro de repository-cgit-configuration]
Sustituye al máximo periodo estadístico predeterminado.
El valor predeterminado es ‘”‘.

repo-string name [parámetro de repository-cgit-configuration]
El valor a mostrar como nombre del repositorio.
El valor predeterminado es ‘”‘.

repo-string owner [parámetro de repository-cgit-configuration]
Un valor usado para identificar a la propietaria del repositorio.
El valor predeterminado es ‘”‘.

repo-string path [parámetro de repository-cgit-configuration]
La ruta absoluta al directorio del repositorio.
El valor predeterminado es ‘”‘.

repo-string readme [parámetro de repository-cgit-configuration]
Una ruta (relativa al repositorio) que especifica un fichero que será incluido literalmente como página "About" de este repositorio.
El valor predeterminado es ‘”‘.

repo-string section [parámetro de repository-cgit-configuration]
Nombre actual de la sección de repositorios - todos los repositorios definidos tras esta opción heredarán el nombre actual de sección.
El valor predeterminado es ‘”‘.

repo-list extra-options [parámetro de repository-cgit-configuration]
Opciones adicionales a agregar al final del fichero cgitrc.
El valor predeterminado es ‘()’.

lista extra-options [parámetro de cgit-configuration]
Opciones adicionales a agregar al final del fichero cgitrc.
El valor predeterminado es ‘()’.

No obstante, puede ser que únicamente desee usar un fichero cgitrc existente. En ese caso, puede proporcionar opaque-cgit-configuration como un registro a cgit-service-type. Como su nombre en inglés indica, una configuración opaca no tiene gran capacidad reflexiva.

Los campos disponibles de opaque-cgit-configuration son:

package cgit [parámetro de opaque-cgit-configuration]
El paquete cgit.

string string [parámetro de opaque-cgit-configuration]
El contenido de cgitrc, como una cadena.

Por ejemplo, si su cgitrc es simplemente la cadena vacía, puede instanciar un servicio cgit de esta manera:

(service cgit-service-type
    (opaque-cgit-configuration
        (cgitrc "")))
Servicio Gitolite

Gitolite (https://gitolite.com/gitolite/) es una herramienta para el almacenamiento de repositorios Git en un servidor central.

Gitolite puede manejar múltiples repositorios y usuarias, y permite una configuración flexible de los permisos de las usuarias sobre los repositorios.

El siguiente ejemplo configuraría Gitolite con la usuaria predeterminada `git` y la clave pública SSH proporcionada.

```
(service gitolite-service-type
  (gitolite-configuration
    (admin-pubkey (plain-file
      "sunombre.pub"
      "ssh-rsa AAAA... guix@example.com"))))
```

Gitolite se configura a través de un repositorio de administración especial que puede clonar, por ejemplo, si configuró Gitolite en `example.org`, ejecutaría la siguiente orden para clonar el repositorio de administración.

```
git clone git@example.com:gitolite-admin
```

Cuando se activa el servicio Gitolite, la clave `admin-pubkey` proporcionada se insertará en el directorio `keydir` en el repositorio `gitolite-admin`. Si esto resultase en un cambio en el repositorio, la revisión se almacenaría con el mensaje “gitolite setup by GNU Guix”.

**gitolite-configuration** [Tipo de datos]

Tipo de datos que representa la configuración de `gitolite-service-type`.

- **package** (predeterminado: `gitolite`)
  - Paquete Gitolite usado.

- **user** (predeterminada: `git`)
  - Usuaria usada por Gitolite. Esta será la usuaria a la que se conectará cuando acceda a Gitolite a través de SSH.

- **group** (predeterminado: `git`)
  - Grupo usado por Gitolite.

- **home-directory** (predeterminado: `"/var/lib/gitolite"`)
  - Directorio en el que se almacenará la configuración y repositorios de Gitolite.

- **rc-file** (predeterminado: `gitolite-rc-file`)
  - Un objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108), que representa la configuración de Gitolite.

- **admin-pubkey** (predeterminada: `#f`)
  - Un objeto “tipo-fichero” (véase Sección 6.7 [Expresiones-G], página 108) usado para la configuración de Gitolite. Se insertará en el directorio `keydir` dentro del repositorio `gitolite-admin`. Para especificar la clave SSH como una cadena, use la función `plain-file`.

```
(plain-file "sunombre.pub" "ssh-rsa AAAA... guix@example.com")
```
8.8.26 Servicios de juegos

El servicio de La batalla por Wesnoth

La batalla por Wesnoth (https://wesnoth.org) es un juego de estrategia táctica, de fantasía y basado en turnos, con varias campañas de una jugadora, y partidas para múltiples jugadoras (tanto en red como localmente).

Variable Scheme wesnothd-service-type

Tipo de servicio para el servicio wesnothd. Su valor debe ser un objeto wesnothd-configuration. Puede instanciarlo de esta manera para ejecutar wesnothd con la configuración predeterminada:

(service wesnothd-service-type)

wesnothd-configuration

Tipo de datos que representa la configuración de wesnothd.

package (predeterminado: wesnoth-server)

El paquete del servidor wesnoth usado.

port (predeterminado: 15000)

Número de puerto usado por el servidor.

8.8.27 Servicio PAM Mount

El módulo (gnu services pam-mount) proporciona un servicio que permite a las usuarias montar volúmenes cuando ingresen al sistema. Debe ser capaz de montar cualquier formato de volumen que el sistema permita.
Variable Scheme ```pam-mount-service-type``` [Variable]

Tipo de servicio para la implementación de PAM Mount.

pam-mount-configuration [Tipo de datos]

Tipo de datos que representa la configuración de PAM Mount.

Toma los siguientes parámetros:

rules Las reglas de configuración que se usarán para generar `/etc/security/pam_mount.conf.xml`.

Las reglas de configuración son elementos SXML (véase Sección “SXML” en [GNU Guile Reference Manual](#)), y las reglas predeterminadas no incluyen el montado de ningún dispositivo para ningún usuario en el ingreso al sistema:

```
'((debug (@ (enable "0"))
  (mntoptions (@ (allow ,(string-join
     ,("nosuid" "nodev" "loop"
     "encryption" "fsck" "nonempty"
     "allow_root" "allow_other")
     ",")))))
 (mntoptions (@ (require "nosuid,nodev")))
 (logout (@ (wait "0")
    (hup "0")
    (term "no")
    (kill "no")))
 (mkmountpoint (@ (enable "1")
    (remove "true"))))
```

Algunos elementos ```volume``` deben añadirse de manera automática para montar volúmenes en el ingreso. El siguiente ejemplo permite a la usuaria ```alicia``` montar su directorio ```HOME``` cifrado y permite al usuario ```rober``` montar la partición donde almacena sus datos:

```
(define reglas-pam-mount
  '((debug (@ (enable "0"))
    (volume (@ (user "alicia")
      (fstype "crypt")
      (path "/dev/sda2")
      (mountpoint "/home/alicia")))))
(volume (@ (user "rober")
  (fstype "auto")
  (path "/dev/sdb3")
  (mountpoint "/home/rober/data")
  (options "defaults,autodefrag,compress")))
    (mntoptions (@ (allow ,(string-join
     ,("nosuid" "nodev" "loop"
     "encryption" "fsck" "nonempty"
     "allow_root" "allow_other")
     ",")))))
 (mntoptions (@ (require "nosuid,nodev"))))
```
8.8.28 Servicios de Guix

Servicio de datos de Guix

El servicio de datos de Guix (http://data.guix.gnu.org) procesa, almacena y proporciona datos acerca de GNU Guix. Esto incluye información sobre paquetes, derivaciones y avisos de “lint”.

Los datos se almacenan en una base de datos PostgreSQL, y están disponibles a través de una interfaz web.

Variable Scheme guix-data-service-type

Tipo de servicio para el servicio de datos de Guix. Su valor debe ser un objeto guix-data-service-configuration. El servicio opcionalmente extiende el servicio getmail, puesto que la lista de correo guix-commit se usa para conocer los cambios del repositorio git de Guix.

guix-data-service-configuration

Tipo de datos que representa la configuración del servicio de datos de Guix.

package (predeterminado: guix-data-service)

El paquete del servicio de datos de Guix usado.

user (usuaria: "guix-data-service")

Usuraria del sistema que ejecuta el servicio.

group (predeterminado: "guix-data-service")

Grupo del sistema que ejecuta el servicio.

port (predeterminado: 8765)

El puerto al que se asociará el servicio web.

host (predeterminada: "127.0.0.1")

El nombre de máquina al que se asociará el servicio web.

getmail-idle-mailboxes (predeterminado: #f)

Si se proporciona un valor, es la lista de bandejas de correo en las cuales la configuración debe indicar su lectura al servicio getmail.
commits-getmail-retriever-configuration (predeterminado: #f)
Si se proporciona un valor, es el objeto getmail-retriever-configuration con el que se configura getmail para obtener recibir el correo de la lista guix-commits.

eextra-options (predeterminadas: '())
Extra command line options for guix-data-service.

eextra-process-jobs-options (default: '())
Extra command line options for guix-data-service-process-jobs.

Servicios de Linux

8.8.29 Servicio Early OOM

Early OOM (https://github.com/rfjakob/earlyoom), también conocido como Earl-yoom, es un daemon minimalista de gestión del llenado de la memoria\(^\text{18}\) que se ejecuta en espacio de usuaria y proporciona una alternativa al gestor del propio núcleo con una respuesta más inmediata y más configurable. Es útil para prevenir que el sistema no responda cuando se queda sin memoria.

earlyoom-service-type [Variable Scheme]
Tipo de servicio para el servicio earlyoom, el daemon Early OOM. Su valor debe ser un objeto earlyoom-configuration, descrito a continuación. El servicio se puede instanciar con su configuración predeterminada de esta manera:

(service earlyoom-service-type)

earlyoom-configuration [Tipo de datos]
Esta es el registro de configuración para el servicio earlyoom-service-type.

earlyoom (predeterminado: earlyoom)
El paquete Earlyoom usado.

minimum-available-memory (predeterminado: 10)
El límite inferior de memoria disponible, en porcentaje.

minimum-free-swap (predeterminado: 10)
El límite inferior de memoria de intercambio libre, en porcentaje.

prefer-regexp (predeterminado: #f)
Una expresión regular (como cadena) que corresponda con los nombres de los procesos que preferiblemente deban pararse.

avoid-regexp (predeterminado: #f)
Una expresión regular (como cadena) que corresponda con los nombres de los procesos que no deban pararse.

memory-report-interval (predeterminado: 0)
Intervalo en segundos con el cual se imprime el informe de memoria. No está activo de manera predeterminada.

\(^{18}\) NdT: Del inglés Out Of Memory.
ignore-positive-oom-score-adj? (predeterminado: #f)
Valor booleano que indica si se realizan ajustes positivos en /proc/*/oom_score_adj.

show-debug-messages? (predeterminado: #f)
Valor booleano que indica si los mensajes de depuración deben imprimirse. Los registros se almacenan en /var/log/earlyoom.log.

send-notification-command (predeterminada: #f)
Puede usarse para proporcionar una orden personalizada para el envío de notificaciones.

8.8.30 Servicios misceláneos

Servicio de huella dactilar
El módulo (gnu services authentication) proporciona un servicio DBus para leer e identificar huellas dactilares mediante un sensor de huellas.

fprintd-service-type [Variable Scheme]
El tipo de servicio para fprintd, que proporciona la capacidad de lectura de huellas dactilares.

(sysctl-service-type)

Servicios de control del sistema
El módulo (gnu services sysctl) proporciona servicios para configurar parámetros del núcleo durante el arranque.

sysctl-service-type [Variable Scheme]
El tipo de servicio para sysctl, que modifica parámetros del núcleo bajo /proc/sys. Para activar el encaminamiento de tráfico IPv4 se puede instanciar de esta manera:

(sysctl-service-type
  (sysctl-configuration
    (settings '((("net.ipv4.ip_forward" . "1"))))))

sysctl-configuration [Tipo de datos]
Tipo de datos que representa la configuración de sysctl.

(sysctl (predeterminado: (file-append procps "/sbin/sysctl")
El ejecutable sysctl usado.

settings (predeterminados: '())
Una lista asociativa que especifica parámetros del núcleo y sus valores.

Servicio del daemon de tarjetas inteligentes PC/SC
El módulo (gnu services security-token) proporciona el siguiente servicio para ejecutar pcsd, el daemon de tarjetas inteligentes PC/SC. pcsd es el programa del daemon para pcsclite y el framework MuscleCard. Es un gestor de recursos que coordina la comunicación con lectores de tarjetas inteligentes, las propias tarjetas y los identificadores criptográficos conectados al sistema.
Capítulo 8: Configuración del sistema

**pcscd-service-type**

Tipo de servicio para el servicio pcscd. Su valor debe ser un objeto pcscd-configuration. Puede instanciarlo de esta manera para ejecutar pcscd con la configuración predeterminada:

\[
\text{(service pcscd-service-type)}
\]

**pcscd-configuration**

Tipo de datos que representa la configuración de pcscd.

- **pcsc-lite** (predeterminado: pcsc-lite)
  El paquete pcsc-lite que proporciona pcscd.

- **usb-drivers** (predeterminado: (list ccid))
  Lista de paquetes que proporcionan los controladores USB para pcscd. Se espera que los controladores se encuentren en el directorio pcsc/drivers del directorio del almacen del paquete.

**Servicio Lirc**

El módulo (gnu services lirc) proporciona el siguiente servicio.

**lirc-service**

Devuelve un servicio que ejecuta LIRC (http://www.lirc.org), un daemon que decodifica señales infrarrojas de dispositivos de control remoto.

De manera opcional, se pueden especificar **device, driver y config-file** (nombre del fichero de configuración) may be specified. Véase el manual de lircd para obtener más detalles.

Por último, **extra-options** es una lista de opciones adicionales para la línea de órdenes proporcionada a lircd.

**Servicio Spice**

El módulo (gnu services spice) proporciona el siguiente servicio.

**spice-vdagent-service**

Devuelve un servicio que ejecuta VDAGENT (https://www.spice-space.org), un daemon que permite compartir el portapapeles con una máquina virtual y la configuración de la resolución de la pantalla de la máquina virtual cuando la ventana de la consola gráfica cambia de tamaño.

**Servicio inputattach**

El servicio inputattach (https://linuxwacom.github.io/) permite el uso de dispositivos de entrada como tabletas Wacom, pantallas táctiles, o joysticks con el servidor gráfico Xorg.

**inputattach-service-type**

Tipo de un servicio que ejecuta inputattach con un dispositivo y reenvía los eventos que produzca.
inputattach-configuration

device-type (predeterminado: "wacom")
Tipo del dispositivo al que conectarse. Ejecute inputattach --help, del paquete inputattach, para ver la lista de tipos de dispositivo implementados.

device (predeterminado: "/dev/ttyS0")
El nombre de fichero para la conexión al dispositivo.

baud-rate (predeterminado: #f)
Tasa de transmisión usada para las conexiones serie. Debe ser un número o #f.

log-file (predeterminado: #f)
Si es verdadero, debe ser el nombre de un fichero en el que registrar los mensajes.

8.8.31 Servicios de diccionario
El módulo (gnu services dict) proporciona el servicio siguiente:

dicod-service [#config (dicod-configuration)]
[Procedimiento Scheme]
Devuelve un servicio que ejecuta el daemon dicod, una implementación del servidor DICT (véase Sección “Dicod” en GNU Dico Manual).

El optional config argument specifies the configuration for dicod, which should be a <dicod-configuration> object, by default it serves the GNU Collaborative International Dictionary of English.

Puede añadir open localhost en su fichero ~/.dico para hacer que localhost sea el servidor predeterminado de su cliente dico (véase Sección “Initialization File” en GNU Dico Manual).

dicod-configuration
[Tipo de datos]
Tipo de datos que representa la configuración de dicod.

dico (predeterminado: dico)
El objeto paquete del servidor de diccionario GNU Dico.

interfaces (predeterminada: '("localhost")')
Es la lista de direcciones IP y puertos, y posiblemente nombres de fichero de sockets, en los que se debe escuchar (véase Sección “Server Settings” en GNU Dico Manual).

handlers (predeterminados: ')()
Lista de objetos <dicod-handler> que identifican los controladores (instancias de módulos).

databases (predeterminada: (list %dicod-database:gcide))
Lista de objetos <dicod-database> que identifican los diccionarios proporcionados.

dicod-handler
[Tipo de datos]
Tipo de datos que representa un controlador de diccionario (instancia de un módulo).

name Nombre del controlador (instancia de un módulo).
module (predeterminado: #f)
Nombre del módulo del controlador de dicod (instancia). Si es #f, el módulo tiene el mismo nombre que el controlador. (véase Sección “Módulos” en GNU Dico Manual).

options Lista de cadenas o expresiones-G que representan los parámetros al módulo de control

dicod-database [Tipo de datos]
Tipo de datos que representa una base de datos de diccionario.

name Nombre de la base de datos, será usada en las órdenes DICT.

handler Nombre del controlador de dicod (instancia de un módulo) usado por esta base de datos (véase Sección “Handlers” en GNU Dico Manual).

complex? (predeterminado: #f)
Determina si se usará la configuración compleja. La configuración compleja necesita un objeto <dicod-handler>, que no es necesario en otro caso.

options Lista de cadenas o expresiones-g que representan los parámetros para la base de datos (véase Sección “Databases” en GNU Dico Manual).

%dicod-database:gcide [Variable Scheme]
Un objeto <dicod-service> que ofrece el diccionario internacional colaborativo de inglés de GNU usando el paquete gcide.

A continuación se encuentra un ejemplo de configuración de dicod-service.

(dicod-service #:config
(dicod-configuration
 (handlers (list (dicod-handler
 (name "wordnet")
 (module "dictorg")
 (options
 (list #~(string-append "dbdir=" #$wordnet))))))
 (databases (list (dicod-database
 (name "wordnet")
 (complex? #t)
 (handler "wordnet")
 (options ’("database=wn")))
 %dicod-database:gcide)))))

Servicio Docker

El módulo (gnu services docker) proporciona los siguientes servicios.

docker-service-type [Variable Scheme]
Este es el tipo del servicio que ejecuta Docker (https://www.docker.com), un daemon que puede ejecutar empaquetados de aplicaciones (a los que a veces nos referimos como “contenedores”) en entornos aislados.
Capítulo 8: Configuración del sistema

docker-configuration

Este es el tipo de datos que representa la configuración de Docker y Containerd.

package (predeterminado: docker)

El paquete de Docker usado.

containerd (predeterminado: containerd)

El paquete Containerd usado.

Servicio Auditd

El módulo (gnu services auditd) proporciona el servicio siguiente:

auditd-service-type

Este es el tipo del servicio que ejecuta auditd (https://people.redhat.com/sgrubb/audit/), un daemon que recolecta información relevante a la seguridad en su sistema.

Ejemplos de cosas que se pueden recolectar:

1. Acceso a ficheros
2. Llamadas al sistema
3. Órdenes invocadas
4. Intentos fallidos de ingreso en el sistema
5. Activaciones de filtros en el cortafuegos
6. Accesos de red

Se puede usar auditctl del paquete audit para añadir o eliminar eventos a recolectar (hasta el siguiente reinicio). Para hacer permanentemente la recolección de dichos eventos, introduzca los parámetros de la línea de órdenes de auditctl en /etc/audit/audit.rules. Se puede usar aureport del paquete audit para ver un informe de todos los eventos registrados. El daemon audit habitualmente registra los eventos en el fichero /var/log/audit.

auditd-configuration

Este es el tipo de datos que representa la configuración de auditd.

audit (predeterminado: audit)

El paquete audit usado.

singularity-service-type

Tipo de servicio que le permite ejecutar Singularity (https://www.sylabs.io/singularity/), una herramienta tipo-Docker para crear y ejecutar aplicaciones empaquetadas (también conocidas como “contenedores”). El valor para este servicio es el paquete de Singularity usado.

El servicio no instala un daemon; en vez de ello, instala programas auxiliares con con el bit setuid de root (véase Sección 8.9 [Programas con setuid], página 389) de modo que usuarias sin privilegios puedan ejecutar singularity run y órdenes similares.
Servicio Nix
El módulo (gnu services nix) proporciona el siguiente servicio.

**nix-service-type**
Tipo del servicio que ejecuta el daemon de construcción del gestor de paquetes Nix (https://nixos.org/nix/). Este es un ejemplo de cómo usarlo:

```
(use-modules (gnu))
(use-service-modules nix)
(use-package-modules package-management)

(operating-system
 ;; ...
 (packages (append (list nix) %base-packages))

 (services (append (list (service nix-service-type)) %base-services)))
```

Tras guix system reconfigure, configure Nix para su usuaria:

- Cree un enlace simbólico a su perfil y active el perfil de Nix:

  ```
  $ ln -s "/nix/var/nix/profiles/per-user/$USER/profile" ~/.nix-profile
  $ source /run/current-system/profile/etc/profile.d/nix.sh
  ```

### 8.9 Programas con setuid

Algunos programas necesitan ejecutarse con privilegios de “root”, incluso cuando se ejecutan por usuarias sin privilegios. Un ejemplo notable es el programa `passwd`, que las usuarias ejecutan para cambiar su contraseña, y que necesita acceso a los ficheros `/etc/passwd` y `/etc/shadow`—algo normalmente restringido a root, por razones de seguridad obvias. Para solventarlo, estos ejecutables tienen **setuid de root**, lo que significa que siempre se ejecutan con privilegios de root (véase Sección “How Change Persona” en The GNU C Library Reference Manual, para más información sobre el mecanismo setuid).

El almacén en sí no puede contener programas setuid: sería un problema de seguridad puesto que cualquier usuaria del sistema puede escribir derivaciones que pueblen el almacén (véase Sección 6.4 [El almacén], página 98). Por tanto, se usa un mecanismo diferente: en vez de cambiar el bit de setuid directamente en los ficheros que se encuentran en el almacén, se permite que la administradora del sistema declare qué programas deberían tener setuid de root.

El campo **setuid-programs** de una declaración `operating-system` contiene una lista de expresiones-G que denotan nombres de programas que tendrán setuid de root (véase Sección 8.1 [Uso de la configuración del sistema], página 161). Por ejemplo, el programa **passwd**, que es parte del paquete Shadow, puede designarse con esta expresión-G (véase Sección 6.7 [Expresiones-G], página 108):

```
#"(string-append #$shadow "/bin/passwd")
```
Un conjunto predeterminado de programas con el bit setuid se define en la variable %setuid-programs del módulo (gnu system).

%setuid-programs [Variable Scheme]

Una lista de expresiones-G que denotan programas comunes que se marcan con setuid de root.

La lista incluye órdenes como passwd, ping, su y sudo.

Para su implementación, los programas con setuid reales se crean en el directorio /run/setuid-programs durante la activación del sistema. Los ficheros en este directorio hacen referencia a los binarios “reales”, que están en el almacén.

8.10 Certificados X.509

En las conexiones HTTPS a servidores Web (esto es, HTTP sobre el mecanismo de seguridad de la capa de transporte, TLS) se envía a los programas clientes un certificado X.509 que el cliente puede usar para autenticar al servidor. Para hacerlo, los clientes verifican que el certificado del servidor está firmado por una de las llamadas autoridades de certificación (AC, CA en inglés). Pero para verificar la firma de una AC, los clientes deben haber obtenido previamente el certificado de dicha AC.

Los navegadores Web como GNU IceCat incluyen su propio conjunto de certificados de AC, de manera que pueden verificar las firmas independientemente.

No obstante, a la mayor parte de otros programas que pueden comunicarse a través de HTTPS—wget, git, w3m, etc.—se les debe informar de dónde pueden encontrar los certificados de CA.

En Guix, esto se lleva a cabo mediante la adición de un paquete que proporcione certificados en el campo packages de la declaración operating-system (véase Sección 8.2 [Referencia de operating-system], página 168). Guix incluye un paquete de este tipo, nss-certs, compuesto por un conjunto de certificados de CA proporcionados como parte de los servicios de seguridad de red de Mozilla (NSS).

Fíjese que no es parte de %base-packages, por lo que debe ser añadido explícitamente. El directorio /etc/ssl/certs, donde la mayor parte de las aplicaciones y bibliotecas buscarán los certificados de manera predeterminada, enlaza a los certificados instalados de manera global.

Las usuarias sin privilegios, incluyendo a usuarias de Guix en una distribución distinta, pueden también instalar su propio paquete de certificados en su perfil. Es necesario definir cierto número de variables de entorno de manera que las aplicaciones y bibliotecas sepan dónde encontrarlos. Por ejemplo, la biblioteca OpenSSL inspecciona las variables SSL_CERT_DIR y SSL_CERT_FILE. Algunas aplicaciones añaden sus variables de entorno propias; por ejemplo, el sistema de control de versiones Git inspecciona el empaquetado de certificados al que apunta la variable de entorno GIT_SSL_CAINFO. Por tanto, en el caso típico se debe ejecutar algo parecido a esto:

$ guix install nss-certs
$ export SSL_CERT_DIR="$HOME/.guix-profile/etc/ssl/certs"
$ export SSL_CERT_FILE="$HOME/.guix-profile/etc/ssl/certs/ca-certificates.crt"
$ export GIT_SSL_CAINFO="$SSL_CERT_FILE"
Como otro ejemplo, R necesita que la variable de entorno `CURL_CA_BUNDLE` apunte al empaquetado de certificados, de manera que se debe ejecutar algo parecido a esto:

```
$ guix install nss-certs
$ export CURL_CA_BUNDLE="$HOME/.guix-profile/etc/ssl/certs/ca-certificates.crt"
```

Para otras aplicaciones puede tener que buscar la variable de entorno necesaria en la documentación relevante.

### 8.11 Selector de servicios de nombres

El módulo `(gnu system nss)` proporciona una interfaz con el fichero de configuración del selector de servicios de nombres o NSS (véase Sección “NSS Configuration File” en The GNU C Library Reference Manual). En resumen, NSS es un mecanismo que permite la extensión de libc con nuevos métodos de búsqueda de “nombres”, lo que incluye nombres de máquinas, nombres de servicios, cuentas de usuario y más (véase Sección “Selector de servicios de nombres” en The GNU C Library Reference Manual).

La configuración de NSS específica, para cada base de datos del sistema, que método de búsqueda debe ser usado, y cómo los varios métodos se enlanzan entre sí—por ejemplo, bajo qué circunstancias NSS deberá probar con el siguiente método en la lista. La configuración de NSS se proporciona en el campo `name-service-switch` de las declaraciones `operating-system` (véase Sección 8.2 [Referencia de operating-system], página 168).

As an example, the declaration below configures the NSS to use the `nss-mdns` back-end ([https://0pointer.de/lennart/projects/nss-mdns/](https://0pointer.de/lennart/projects/nss-mdns/)), which supports host name lookups over multicast DNS (mDNS) for host names ending in `.local`:

```
(name-service-switch
 (hosts (list %files ;primero, comprueba /etc/hosts

 ;; Si lo anterior no funcionó, prueba
 ;; con 'mdns_minimal'.
 (name-service
  (name "mdns_minimal")

 ;; 'mdns_minimal' tiene autoridad sobre
 ;; '.local'. Cuando devuelve 'not-found,
 ;; no es necesario intentarlo con los
 ;; métodos siguientes.
  (reaction (lookup-specification
   (not-found => return))))

 ;; Si no, usa DNS.
 (name-service
  (name "dns"))

 ;; Finalmente, prueba con 'mdns' "al completo".
 (name-service
  (name "mdns"))))
```
No se preocupe: la variable `%mdns-host-lookup-nss` (véase a continuación) contiene esta configuración, de manera que no tiene que escribirla si todo lo que desea es que funcione la búsqueda de nombres de máquina en `.local`.

Fíjese que, en este caso, además de establecer el valor de `name-service-switch` en la declaración `operating-system`, es necesario también usar el servicio `avahi-service-type` (véase Sección 8.8.4 [Servicios de red], página 200) o `%desktop-services`, donde está incluido. Esto permite el acceso a `nss-mdsn` desde el daemon de la caché del servicio de nombres (véase Sección 8.8.1 [Servicios base], página 183).

Por conveniencia, las siguientes variables proporcionan configuraciones NSS típicas.

`%default-nss` [Variable Scheme]

Esta es la configuración predeterminada del selector de servicios de nombres, un objeto `name-service-switch`.

`%mdns-host-lookup-nss` [Variable Scheme]

Esta es la configuración del selector de servicios de nombres que permite la búsqueda de nombres de máquinas por DNS multicast (mDNS) para nombres de máquinas terminados en `.local`.

La referencia de la configuración del selector de servicios de nombres se proporciona a continuación. Tiene una asociación directa con el formato del fichero de configuración de la biblioteca C, por lo que se recomienda el manual de la biblioteca C para obtener más información (véase Sección “NSS Configuration File” en *The GNU C Library Reference Manual*). En comparación con el formato del fichero de configuración del NSS de libc, no solo tiene solo la ventaja de la cálida sensación proporcionada por la adición de paréntesis que tanto nos gustan, sino que también tiene comprobaciones estáticas: conocerá los errores sintácticos y tipográficos con la ejecución de `guix system`.

`name-service-switch` [Tipo de datos]

El tipo de datos que representa la configuración del selector de servicios de nombres (NSS). Cada campo a continuación representa una de las bases de datos del sistema admitidas.

- aliases
- ethers
- group
- gshadow
- hosts
- initgroups
- netgroup
- networks
- password
- public-key
- rpc
- services
- shadow

Las bases de datos del sistema que maneja el NSS. Cada uno de estos campos debe ser una lista de objetos `<name-service>` (véase a continuación).
Capítulo 8: Configuración del sistema

name-service
Este es el tipo de datos que representa un servicio de nombres real y la acción de búsqueda asociada.

name
Una cadena que denota el nombre de servicio (véase Sección “Services in the NSS configuration” en The GNU C Library Reference Manual).

reaction
Una acción especificada mediante el uso del macro lookup-specification (véase Sección “Actions in the NSS configuration” en The GNU C Library Reference Manual). Por ejemplo:

\[
\text{(lookup-specification (unavailable => continue) (success => return))}
\]

8.12 Disco en RAM inicial

Para el propósito del arranque inicial, se le proporciona al núcleo Linux-libre un disco inicial de RAM, o initrd. Un initrd contiene un sistema de ficheros raíz temporal así como un guión de inicialización. Este último es responsable del montaje del sistema de ficheros raíz real, así como de la carga de cualquier módulo del núcleo que pueda ser necesario para esta tarea.

El campo initrd-modules de una declaración operating-system le permite especificar qué módulos del núcleo Linux-libre deben estar disponibles en el initrd. En particular, aquí es donde se debe enumerar los módulos que controlen realmente el disco duro donde su partición raíz se encuentre—aunque el valor predeterminado de initrd-modules debería cubrir la mayor parte de casos de uso. Por ejemplo, en caso de necesitar el módulo megaraid_sas además de los módulos predeterminados para poder acceder a sistema de ficheros raíz, se podría escribir:

\[
\text{(operating-system ;; ...}
\text{(initrd-modules (cons "megaraid_sas" %base-initrd-modules))})
\]

%base-initrd-modules
Esta es la lista de módulos del núcleo que se incluyen en el initrd predeterminado.

Más allá, si necesita personalizaciones de un nivel más bajo, el campo initrd de una declaración operating-system le permite especificar qué initrd desea usar. El módulo (gnu system linux-initrd) proporciona tres formas de construir un initrd: el procedimiento de alto nivel base-initrd y los procedimientos de bajo nivel raw-initrd y expression->initrd.

El procedimiento base-initrd está pensado para cubrir la mayor parte de usos comunes. Por ejemplo, si desea añadir algunos módulos del núcleo que deben cargarse durante el arranque, puede definir el campo initrd de la declaración de sistema operativo de esta forma:

\[
\text{(initrd (lambda (sistemas-de-ficheros . resto))}
\]
El procedimiento base-initrd también maneja casos de uso comunes que implican el uso del sistema en un anfitrión QEMU, o como un sistema “live” con un sistema de ficheros raíz volátil.

El procedimiento base-initrd se construye sobre el procedimiento raw-initrd. Al contrario que base-initrd, raw-initrd no funciona a alto nivel, como sería intentar deducir qué módulos del núcleo y paquetes deben incluirse en el initrd. Un ejemplo de uso de raw-initrd es cuando una usuaria tiene personalizada una configuración del núcleo Linux y los módulos predeterminados del núcleo que incluye base-initrd no están disponibles.

El disco inicial de RAM producido por base-initrd o raw-initrd inspecciona varias opciones proporcionadas por la línea de órdenes al núcleo Linux (esto es, argumentos pasados a través de la orden linux de GRUB, o de la opción -append de QEMU), notablemente:

--load=arranque
Indica al disco de RAM inicial que cargue arranque, un fichero que contiene un programa Scheme, una vez haya montado el sistema de ficheros raíz.
Guix usa esta opción para proporcionar el control a un programa de arranque que ejecuta los programas de activación de servicios y lanza GNU Shepherd, el sistema de inicialización.

--root=raíz
Monta raíz como el sistema de ficheros raíz. raíz puede ser un nombre de dispositivo como /dev/sda1, una etiqueta del sistema de ficheros o un UUID del sistema de ficheros.

--system=sistema
Hace que /run/booted-system y /run/current-system apunten a sistema.

modprobe.blacklist=módulos...
Indica al disco inicial de RAM así como a la orden modprobe (del paquete kmod) que deben negarse a cargar módulos. módulos debe ser una lista separada por comas de nombres de módulos—por ejemplo, usbkbd,9pnet.

--repl

Una vez conocidas todas las características que proporcionan los discos iniciales de RAM que producen base-initrd y raw-initrd, a continuación veremos cómo usarlas y personalizarlos más aún.

Devuelve una derivación que construye una imagen de inicio en RAM (initrd) en crudo. sistemas-de-ficheros es una lista de sistemas de ficheros que serán montados por initrd, posiblemente en adición al sistema de ficheros raíz especificado en la línea de órdenes del núcleo vía --root. linux-modules es una lista de módulos del núcleo a ser cargados en tiempo de arranque. mapped-devices es una lista de dispositivos asociados a realizar antes de montar sistemas-de-ficheros (véase Sección 8.4 [Dispositivos traducidos], página 175). helper-packages es una lista de paquetes a ser copiados en el initrd. Puede incluir e2fsck/static o otros paquetes necesarios en la imagen de RAM para comprobar el sistema de ficheros raíz.

Cuando su valor es verdadero, keyboard-layout es un registro <keyboard-layout> que denota la distribución de teclado en consola deseada. Esto se realiza previamente a que los dispositivos configurados en mapped-devices se inicien y antes de que los sistemas de ficheros en file-systems se monten, de manera que, en caso de que la usuaria tuviese que introducir una contraseña o usar la sesión interactiva, esto suceda usando la distribución de teclado deseada.

Cuando qemu-networking? es verdadero, configura la red con los parámetros QEMU estándar. Cuando virtio? es verdadero, carga módulos adicionales para que la imagen en RAM pueda ser usada como un sistema virtualizado por QEMU con controladores paravirtualizados de E/S.

Cuando volatile-root? es verdadero, el sistema de ficheros raíz tiene permisos de escritura pero cualquier cambio realizado se perderá.

base-initrd sistemas-de-ficheros [Procedimiento Scheme]

[#:mapped-devices ()] [#:keyboard-layout #f] [#:qemu-networking? #f]
[#:volatile-root? #f] [#:linux-modules ()]

Devuelve como un objeto tipo-fichero una imagen de inicio en RAM genérica, con los módulos del núcleo tomados de linux. sistemas-de-ficheros es una lista de sistemas de ficheros listos para ser montados por la imagen, posiblemente en adición al sistema de ficheros raíz especificado en la línea de órdenes del núcleo vía --root. mapped-devices es una lista de asociación de dispositivos a realizar antes de montar los sistemas-de-ficheros.

Cuando su valor es verdadero, keyboard-layout es un registro <keyboard-layout> que denota la distribución de teclado en consola deseada. Esto se realiza previamente a que los dispositivos configurados en mapped-devices se inicien y antes de que los sistemas de ficheros en file-systems se monten, de manera que, en caso de que la usuaria tuviese que introducir una contraseña o usar la sesión interactiva, esto suceda usando la distribución de teclado deseada.


El initrd incorpora automáticamente todos los módulos del núcleo necesarios para sistemas-de-ficheros y para las opciones proporcionadas. Módulos del núcleo adicionales pueden proporcionarse a través de linux-modules. Se añadirán al initrd y se cargarán en tiempo de arranque en el orden que aparezcan.
No es necesario decir que los initrd que producimos y usamos embebemos un Guile enlazado estáticamente, y que el programa de inicialización es un programa Guile. Esto proporciona mucha flexibilidad. El procedimiento `expression->initrd` construye un initrd de ese tipo, una vez proporcionado el programa a ejecutar en dicho initrd.

```scheme
(expression->initrd exp [#:guile %guile-static-stripped] [#:name "guile-initrd"]
```

Devuelve como un objeto tipo-fichero el initrd de Linux (un archivador cpio comprimido con gzip) que contiene guile y que evalúa a exp, una expresión-G, al arranque. Todas las derivaciones a las que exp hace referencia se copian automáticamente en el initrd.

### 8.13 Configuración del gestor de arranque

El sistema operativo permite varios cargadores de arranque. El cargador de arranque se configura mediante el uso de la declaración `bootloader-configuration`. Todos los campos de esta estructura son independientes del cargador de arranque excepto uno, `bootloader`, que indica el cargador de arranque a configurar e instalar.

Algunos de los cargadores de arranque no inspeccionan todos los campos de `bootloader-configuration`. Por ejemplo, el cargador de arranque extlinux no permite temas y por lo tanto ignora el campo `theme`.

```scheme
bootloader-configuration
```

El tipo de una declaración de configuración del cargador de arranque.

- **bootloader**
  - El cargador de arranque usado, como un objeto `bootloader`. De momento se aceptan `grub-bootloader`, `grub-efi-bootloader`, `extlinux-bootloader` y `u-boot-bootloader`.
  - `grub-efi-bootloader` permite el arranque en sistemas modernos que usan la interfaz extendida de firmware unificada (UEFI). Es el que debería ser usado si la imagen de instalación contiene un directorio `/sys/firmware/efi` cuando la arranca en su sistema.
  - `grub-bootloader` permite el arranque en máquinas basadas en Intel en modo “antiguo” BIOS.
  - Los cargadores de arranque se describen en los módulos (`gnu bootloader ...`). En particular, (`gnu bootloader u-boot`) contiene definiciones de cargadores de arranque para un amplio rango de sistemas ARM y AArch64, mediante el uso del cargador de arranque U-Boot (https://www.denx.de/wiki/U-Boot/).

- **target**
  - Una cadena que indica donde se instalará el cargador de arranque.
  - La interpretación depende del cargador de arranque en cuestión. Para `grub-bootloader`, por ejemplo, debe ser un nombre de dispositivo que entienda la orden `install` del cargador de arranque, como `/dev/sda` o `(hd0)` (véase Sección “Invoking grub-install” en GNU GRUB Manual).
  - Para `grub-efi-bootloader`, debe apuntar al punto de montaje del sistema de ficheros EFI, habitualmente `/boot/efi`. 
Capítulo 8: Configuración del sistema

menu-entries (predeterminadas: ()

Una lista posiblemente vacía de objetos menu-entry (véase a continuación), que indican entradas que deben aparecer en el menú del cargador de arranque, además de la entrada del sistema actual y la entrada que apunta a generaciones previas del sistema.

default-entry (predeterminada: 0)

El índice de la entrada del menú de arranque por omisión. El índice 0 es para la entrada del sistema actual.

timeout (predeterminado: 5)

El número de segundos que se esperará entrada por el teclado antes de arrancar. El valor 0 indica que se debe arrancar de forma inmediata, y -1 que se debe esperar indefinidamente.

keyboard-layout (predeterminada: #f)

Si es #f, el menú del cargador de arranque (si existe) usa la distribución de teclado predeterminada, habitualmente inglés estadounidense (“qwerty”).

En otro caso, debe ser un objeto keyboard-layout (véase Sección 8.6 [Distribución de teclado], página 179).

Nota: Esta opción se ignora actualmente por todos los cargadores de arranque menos grub y grub-efi.

theme (predeterminado: #f)

El objeto del tema del cargador de arranque que describe el tema usado. Si no se proporciona ningún tema, algunos cargadores de arranque pueden usar un tema por omisión, lo cual es cierto en GRUB.

terminal-outputs (predeterminadas: ’(gfxterm))

Los terminales de salida que se usarán para el menú de arranque, como una lista de símbolos. GRUB acepta los valores: console, serial, serial_{0-3}, gfxterm, vga_text, mda_text, morse y pkmodem. Este campo corresponde con la variable GRUB_TERMINAL_OUTPUT (véase Sección “Simple configuration” en GNU GRUB manual).

terminal-inputs (predeterminadas: ’(())

Los terminales de entrada que se usarán para el menú de arranque, como una lista de símbolos. Para GRUB, el valor predeterminado es el terminal nativo de la plataforma determinado en tiempo de ejecución. GRUB acepta los valores: console, serial, serial{0-3}, at_keyboard y usb_keyboard. Este campo corresponde a la variable GRUB GRUB_TERMINAL_INPUT (véase Sección “Simple configuration” en GNU GRUB manual).

serial-unit (predeterminada: #f)

La unidad serie usada por el cargador de arranque, como un entero del 0 al 3. Para GRUB, se selecciona en tiempo de ejecución; actualmente GRUB selecciona 0 lo que corresponde a COM1 (véase Sección “Serial terminal” en GNU GRUB manual).
serial-speed (predeterminada: #f)
La velocidad de la interfaz serie, como un entero. Para GRUB, el valor predeterminado se selecciona en tiempo de ejecución, actualmente GRUB selecciona 9600 bps (véase Sección “Serial terminal” en GNU GRUB manual).

Si deseas listar entradas adicionales para el menú de arranque a través del campo menu-entries mostrado previamente, deberás crearlas con la forma menu-entry. Por ejemplo, imagine que desea ser capaz de arrancar otra distribución (¡difícil de imaginar!), puede definir una entrada de menú de esta forma:

(menu-entry
  (label "La otra distribución")
  (linux "/boot/old/vmlinux-2.6.32")
  (linux-arguments '("root=/dev/sda2"))
  (initrd "/boot/old/initrd")
)

Los detalles se encuentran a continuación.

menu-entry [Tipo de datos]
El tipo de una entrada en el menú del cargador de arranque.

label La etiqueta a mostrar en el menú—por ejemplo, "GNU".

linux La imagen del núcleo Linux a arrancar, por ejemplo:

  (file-append linux-libre "/bzImage")

Con GRUB, también es posible especificar un dispositivo explícitamente mediante el uso de la convención de nombres de dispositivo de GRUB (véase Sección “Naming convention” en GNU GRUB manual), por ejemplo:

  "(hd0,msdos1)/boot/vmlinuz"

Si se especifica el dispositivo explícitamente como en el ejemplo anterior, el campo device se ignora completamente.

linux-arguments (predeterminados: ()
La lista de parámetros extra de línea de órdenes para el núcleo Linux—por ejemplo, ("console=ttys0").

initrd Una expresión-G o una cadena que contiene el nombre de fichero del disco inicial en RAM usado (véase Sección 6.7 [Expresiones-G], página 108).

device (predeterminado: #f)
El dispositivo donde se encuentran el núcleo y el initrd—es decir, para GRUB, raíz de esta entrada de menú (véase Sección “root” en GNU GRUB manual).

Puede ser una etiqueta de sistema de ficheros (una cadena), un UUID de sistema de ficheros (un vector de bytes, véase Sección 8.3 [Sistemas de ficheros], página 172), o #f, en cuyo caso el cargador de arranque buscará el dispositivo que contenga el fichero especificado por el campo linux (véase Sección “search” en GNU GRUB manual). No debe ser un nombre de dispositivo del SO como /dev/sda1.
De momento únicamente GRUB permite el uso de temas. Los temas de GRUB se crean mediante el uso de `grub-theme`, todavía no documentado completamente.

`grub-theme` [Tipo de datos]
Tipo de datos que representa la configuración de un tema de GRUB.

`gfxmode` (predeterminado: `'("auto")`
El modo gráfico `gfxmode` de GRUB configurado (una lista de cadenas con resoluciones de pantalla, véase Sección “gfxmode” en GNU GRUB manual).

`%default-theme` [Variable Scheme]
Este es el tema predeterminado de GRUB que usa el sistema operativo si no se especifica el campo `theme` en el registro `bootloader-configuration`.

Viene con una bonita imagen de fondo que muestra los logos de GNU y Guix.

Por ejemplo, para usar una resolución distinta de la predeterminada, puede usar algo como esto:

```
(bootloader
 (grub-configuration
 ;; ...
 (theme (grub-theme
 (inherit %default-theme)
 (gfxmode '"(1024x786x32" "auto"))))))
```

8.14 Invocación de `guix system`

Una vez haya escrito la declaración de sistema operativo como se ha visto en la sección previa, puede instanciarse mediante el uso de la orden `guix system`. Su sinopsis es:

```
guix system opciones... accion fichero
```

`fichero` debe ser el nombre de un fichero que contenga una declaración `operating-system`. `accion` especifica cómo se instancia el sistema operativo. Actualmente se permiten los siguientes valores:

`search` Muestra las definiciones de tipos de servicio disponibles que corresponden con las expresiones regulares proporcionadas, ordenadas por relevancia:

```
$ guix system search console
name: console-fonts
location: gnu/services/base.scm:806:2
extends: shepherd-root
description: Install the given fonts on the specified ttys (fonts are per + virtual console on GNU/Linux). The value of this service is a list of + tty/font pairs. The font can be the name of a font provided by the ‘kbd’ + package or any valid argument to ‘setfont’, as in this example: + + '("tty1" . "LatGrkCyr-8x16") + + ('"tty2" . (file-append + + font-tamzen
```
Como con `guix package --search`, el resultado se obtiene en formato `recutils`, lo que facilita el filtrado de la salida (véase `GNU recutils manual`).

**reconfigure**

Construye el sistema operativo descrito en `fichero`, lo activa, y se constituye como estado actual.\(^{19}\)

Nota: Es altamente recomendable ejecutar `guix pull` antes de la primera ejecución de `guix system reconfigure` (véase Sección 4.6 [Invocación de guix pull], página 51). No hacerlo puede ocasionar que se obtenga una versión más antigua de Guix una vez que `reconfigure` se haya completado.

Lleva a efecto toda la configuración especificada en `fichero`: cuentas de usuario, servicios del sistema, lista de paquetes global, programas con setuid, etc. La orden inicia los servicios del sistema especificados en `fichero` que no estén actualmente en ejecución; si un servicio se encuentra en ejecución esta orden prepara su actualización durante la próxima parada (por ejemplo, con `herd stop X` o `herd restart X`).

Esta orden crea una nueva generación cuyo número es el sucesor de la siguiente generación (como lo muestra `guix system list-generations`). Si esa generación ya existe, será sobreescrita. Este comportamiento es el mismo que el de `guix package` (véase Sección 4.2 [Invocación de guix package], página 36).

También añade una entrada al cargador de arranque para la nueva configuración del sistema operativo—en caso de que no se proporcione la opción `--no-bootloader`. Con GRUB, mueve las entradas de configuraciones antiguas a

---

\(^{19}\) Esta acción (y las acciones relacionadas `switch-generation` y `roll-back`) son usables únicamente en sistemas que ya ejecuten el sistema Guix.
un submenú, permitiendo la selección de una generación previa del sistema en tiempo de arranque en caso necesario.

Tras la finalización, el nuevo sistema se despliega en /run/current-system. Este directorio contiene metadatos de procedencia: la lista de canales usados (véase Sección 4.7 [Canales], página 54) y fichero en sí, cuando esté disponible. Esta información es útil en caso de que desee inspeccionar posteriormente cómo se construyó está generación en particular.

De hecho, asumiendo que fichero está autocontenido, puede construir de nuevo la generación $n$ de su sistema operativo con:

```bash
guix time-machine \
  -C /var/guix/profiles/system-n-link/channels.scm -- \
  system reconfigure \
  /var/guix/profiles/system-n-link/configuration.scm
```

¡Puede pensar en ello como una especie de control de versiones incorporado en Guix! Su sistema no es únicamente un artefacto binario: transporta sus propias fuentes con él. Véase Sección 8.17.3 [Referencia de servicios], página 416, para más información sobre el seguimiento de procedencia.

**switch-generation**

Switch to an existing system generation. This action atomically switches the system profile to the specified system generation. It also rearranges the system’s existing bootloader menu entries. It makes the menu entry for the specified system generation the default, and it moves the entries for the other generations to a submenu, if supported by the bootloader being used. The next time the system boots, it will use the specified system generation.

El cargador de arranque en sí no se reinstala durante esta orden. Por tanto, el cargador de arranque instalado se usa con un fichero de configuración actualizado.

La generación deseada puede especificarse explícitamente con su número de generación. Por ejemplo, la siguiente invocación cambiaría a la generación 7 del sistema:

```
guix system switch-generation 7
```

La generación deseada puede especificarse también de forma relativa a la generación actual con la forma $+N$ o $-N$, donde $+3$ significa “3 generaciones después de la generación actual”, y $-1$ significa “1 generación antes de la generación actual”. Cuando se especifica un valor negativo como $-1$ debe ir precedido de $--$ para evitar que se analice como una opción. Por ejemplo:

```
guix system switch-generation -- -1
```

Actualmente, el efecto de la invocación de esta acción es únicamente cambiar el perfil del sistema a una generación existente y redistribuir las entradas del menú de arranque. Para realmente empezar a usar la generación deseada del sistema, debe reiniciar tras esta acción. En el futuro, se actualizará para hacer lo mismo que reconfigure, como activación y desactivación de servicios.

Esta acción fallará si la generación especificada no existe.
Capítulo 8: Configuración del sistema

roll-back
Cambia a la generación de sistema previa. Tras el siguiente arranque del sistema, usará la generación de sistema precedente. Es la operación inversa de reconfigure, y es equivalente a la invocación de switch-generation con -1 como parámetro.

Actualmente, como con switch-generation, debe reiniciar tras la ejecución de esta acción para realmente empezar a usar la generación de sistema precedente.

delete-generations
Elimina generaciones del sistema, haciendo posible su recolección con la basura (véase Sección 4.5 [Invocación de guix gc], página 48, para información sobre cómo llevar a cabo la “recolección de basura”).

Esto funciona del mismo modo que guix package --delete-generations (véase Sección 4.2 [Invocación de guix package], página 36). Sin parámetros, se eliminan todas las generaciones del sistema excepto la actual:

    guix system delete-generations

También puede seleccionar las generaciones que desea eliminar. El siguiente ejemplo elimina todas las generaciones del sistema que tienen más de dos meses de antigüedad:

    guix system delete-generations 2m

La ejecución de esta orden reinstala automáticamente el cargador de arranque con una lista de entradas del menú actualizada—por ejemplo, el submenú de generaciones antiguas en GRUB no mostrará las generaciones que hayan sido borradas.

build
Construye la derivación del sistema operativo, que incluye todos los ficheros de configuración y programas necesarios para el arranque y la ejecución del sistema. Esta acción no instala nada en realidad.

init
Construye el directorio dado con todos los ficheros necesarios para ejecutar el sistema operativo especificado en fichero. Esto es útil para la instalación inicial de Guix. Por ejemplo:

    guix system init mi-configuración-del-so.scm /mnt

copia a /mnt todos los elementos del almacén necesarios para la configuración especificada en mi-configuración-del-so.scm. Esto incluye los ficheros de configuración, paquetes y demás. También crea otros ficheros esenciales necesarios para la correcta operación del sistema—por ejemplo, los directorios /etc, /var y /run, y el fichero /bin/sh.

Esta orden también instala el cargador de arranque en el destino especificado en mi-conf-del-so.scm, siempre que no se proporcione la opción --no-bootloader.

vm
Construye una máquina virtual que contiene el sistema operativo declarado en fichero, y devuelve un guión que ejecuta dicha máquina virtual (VM).

Nota: La acción vm y otras presentadas a continuación pueden usar la funcionalidad KVM del núcleo Linux-libre. Específicamente, si la máquina permite la virtualización hardware, debe cargarse el
correspondiente módulo KVM del núcleo, debe existir el nodo del dispositivo /dev/kvm y tanto la propia usuaria como las usuarias de construcción del daemon deben tener acceso de lectura y escritura al mismo (véase Sección 2.4.1 [Configuración del entorno de construcción], página 9).

Los parámetros proporcionados al guión se pasan a QEMU como en el siguiente ejemplo, que activa la red y solicita 1 GiB de RAM para la máquina emulada:

```
$ /gnu/store/...-run-vm.sh -m 1024 -smp 2 -net user,model=virtio-net-pci
```

La VM comparte su almacen con el sistema anfitrión.

Sistemas de ficheros adicionales pueden compartirse entre la máquina anfitriona y la virtual mediante el uso de las opciones --share y --expose: la primera especifica un directorio a compartir con acceso de escritura, mientras que la última proporciona solo acceso de lectura al directorio compartido.

El siguiente ejemplo crea una máquina virtual en la que el directorio de la usuaria es accesible en modo solo-lecture, y donde el directorio /intercambio está asociado de forma lectura-escritura con $HOME/tmp en el sistema anfitrión:

```
guix system vm mi-configuración.scm \
  --expose=$HOME --share=$HOME/tmp=/intercambio
```

En GNU/Linux, lo predeterminado es arrancar directamente el núcleo; esto posee la ventaja de no necesitar únicamente una pequeña imagen del disco raíz pequeña ya el el almacen de la anfitriona puede montarse.

La opción --full-boot fuerza una secuencia de arranque completa, desde el cargador de arranque. Esto necesita más espacio en disco ya que la imagen raíz que contiene el núcleo, initrd y los ficheros de datos del cargador de arranque deben crearse. La opción --image-size puede usarse para especificar el tamaño de la imagen.

```
vm-image
disk-image
docker-image
```

Devuelve una máquina virtual, imagen de disco o imagen Docker del sistema operativo declarado en fichero que es independiente. Por omisión, guix system estima el tamaño de la imagen necesario para almacenar el sistema, pero puede usar la opción --image-size para especificar un valor. Las imágenes Docker se construyen para que contengan exactamente lo que necesitan, por lo que la opción --image-size se ignora en el caso de docker-image.

Puede especificar el sistema de ficheros raíz mediante el uso de la opción --file-system-type. Su valor predeterminado es ext4.

Cuando se usa vm-image, la imagen devuelta está en formato qcow2, que QEMU puede usar eficientemente. Véase Sección 8.16 [Ejecutar Guix en una máquina virtual], página 410, para más información sobre cómo ejecutar la imagen en una máquina virtual.

Con disk-image se produce una imagen de disco cruda; puede copiarse tal cual en una memoria USB, por ejemplo. Asumiendo que /dev/sdc es el dispositivo
que corresponde a la memoria USB, se podría copiar la imagen con la siguiente orden:

```bash
# dd if=$(guix system disk-image mi-so.scm) of=/dev/sdc
```

Con `docker-image` se produce una imagen Docker. Guix construye la imagen de cero, no de una imagen Docker base preexistente. Como resultado, contiene exactamente lo definido en el fichero de configuración del sistema operativo. Puede cargar la imagen y ejecutar un contenedor Docker mediante el uso de ordenes como las siguientes:

```bash
id_imagen='docker load < guix-system-docker-image.tar.gz'

id_contenedor='docker create $id_imagen'

docker start $id_contenedor
```

Esta orden arranca un contenedor Docker nuevo a partir de la imagen especificada. El sistema Guix se arrancará de la manera habitual, lo que implica el inicio de cualquier servicio que se haya definido en la configuración del sistema operativo. Puede iniciar una sesión de shell interactiva en el contenedor mediante el uso de `docker exec`:

```bash
docker exec -ti $container_id /run/current-system/profile/bin/bash --login
```

Dependiendo de lo que ejecute en el contenedor Docker, puede ser necesario proporcionar permisos adicionales al contenedor. Por ejemplo, si pretende construir software mediante el uso de Guix dentro del contenedor Docker, puede tener que proporcionar la opción `--privileged` a `docker create`.

container

Devuelve un guión de la ejecución del sistema operativo declarado en fichero dentro de un contenedor. Los contenedores son un conjunto de mecanismos de aislamiento ligeros que proporciona el núcleo Linux-libre. Los contenedores necesitan sustancialmente menos recursos que máquinas virtuales completas debido a que el núcleo, los objetos compartidos y otros recursos pueden compartirse con el sistema anfitrión; esto también significa que proporcionan un menor aislamiento.

En este momento, el guión debe ejecutarse como root para permitir más de una única usuaria y grupo. El contenedor comparte su almacén con la máquina anfitrión.

Como con la acción `vm` (véase [guix system vm], página 402), sistemas de ficheros adicionales a compartir entre la máquina anfitrión y el contenedor pueden especificarse mediante el uso de las opciones `--share` y `--expose`:

```bash

guix system container mi-configuración.scm \
--expose=$HOME --share=$HOME/tmp=/intercambio
```

Nota: Esta opción requiere Linux-libre 3.19 o posterior.

opciones puede contener cualquiera de las opciones de construcción comunes (véase Sección 7.1.1 [Opciones comunes de construcción], página 119). Además, opciones puede contener una de las siguientes:

`--expression=expr`

Considera el sistema operativo al cual evalúa `expr`. Es una alternativa a la especificación de un fichero que evalúa a un sistema operativo. Se usa para la
generación de la imagen de instalación de Guix (véase Sección 3.9 [Construcción de la imagen de instalación], página 34).

--system=sistema
-s sistema
Intenta la construcción para sistema en vez de para el tipo de la máquina anfitriona. Funciona como en guix build (véase Sección 7.1 [Invocación de guix build], página 119).

--derivation
-d Devuelve el nombre de fichero de la derivación del sistema operativo proporcionado sin construir nada.

--save-provenance
Como se ha mostrado previamente, guix system init y guix system reconfigure siempre almacenan información de procedencia a través de un servicio dedicado (véase Sección 8.17.3 [Referencia de servicios], página 416). No obstante, otras órdenes no hacen esto de manera predeterminada. Si desea, digamos, crear una imagen de máquina virtual que contenga información de procedencia, puede ejecutar:

    guix system vm-image --save-provenance config.scm

De este modo, la imagen resultante “embeberá sus propias fuentes” de manera efectiva en forma de metadatos en /run/current-system. Con dicha información se puede reconstruir la imagen para asegurarse de que realmente contiene lo que dice contener; o se puede usar para derivar una variante de la imagen.

--file-system-type=tipo
-t tipo Para la acción disk-image, crea un sistema de ficheros del tipo proporcionado en la imagen.
Cuando se omite esta opción, guix system usa ext4.
--file-system-type=iso9660 produce una imagen ISO-9660, que puede ser grabada en CD y DVD.

--image-size=tamaño
Junto a las acciones vm-image y disk-image, crea una imagen del tamaño proporcionado. tamaño debe ser un número de bytes o puede incluir una unidad como sufijo (véase Sección “Block size” en GNU Coreutils).
Cuando se omite esta opción, guix system calcula una estimación del tamaño de la imagen en función del tamaño del sistema declarado en fichero.

--network
-N Con la acción container, permite a los contenedores acceder a la red de la máquina anfitriona, es decir, no crea un espacio de nombres de red.

--root=fichero
-r fichero
Hace que fichero sea un enlace simbólico al resultado, y lo registra como una raíz del recolector de basura.

--skip-checks
Omite las comprobaciones de seguridad previas a la instalación.
Por omisión, `guix system init` y `guix system reconfigure` realizan comprobaciones de seguridad: se aseguran de que los sistemas de ficheros que aparecen en la declaración `operating-system` realmente existen (véase Sección 8.3 [Sistemas de ficheros], página 172) y que cualquier módulo del núcleo Linux que pudiese necesitarse durante el arranque se encuentre en `initrd-modules` (véase Sección 8.12 [Disco en RAM inicial], página 393). El uso de esta opción omite todas estas comprobaciones.

`--on-error=estrategia`

Aplica `estrategia` cuando ocurre un error durante la lectura de `fichero`. `estrategia` puede ser uno de los siguientes valores:

- `nothing-special`:
  Informa concisamente del error y termina la ejecución. Es la estrategia predeterminada.

- `backtrace`:
  Del mismo modo, pero también muestra la secuencia de llamadas.

- `debug`:

Una vez haya construido, configurado, reconfigurado y re-reconfigurado su instalación de Guix, puede encontrar útil enumerar las generaciones del sistema operativo disponibles en el disco—y que puede seleccionar en el menú de arranque:

- `describe`:
  Describe la generación actual del sistema: su nombre de fichero, el núcleo y el cargador de arranque usados, etcétera, así como información de procedencia cuando esté disponible.

- `list-generations`:
  Muestra un resumen de cada generación del sistema operativo disponible en el disco, de manera legible por humanos. Es similar a la opción `--list-generations` de `guix package` (véase Sección 4.2 [Invocación de guix package], página 36).

  De manera opcional, se puede especificar un patrón, con la misma sintaxis que la usada en `guix package --list-generations`, para restringir la lista de generaciones mostradas. Por ejemplo, la siguiente orden muestra generaciones que tienen hasta 10 días de antigüedad:

  ```
  $ guix system list-generations 10d
  ```

  ¡La orden `guix system` tiene aún más que ofrecer! Las siguientes órdenes le permiten visualizar cual es la relación entre los servicios del sistema:

- `extension-graph`:
  Emite en formato Dot/Graphviz por la salida estándar el grafo de extensiones de servicio del sistema operativo definido en `fichero` (véase Sección 8.17.1 [Com-
Capítulo 8: Configuración del sistema

posición de servicios], página 412, para más información sobre extensiones de servicio).
La orden:

```
$ guix system extension-graph fichero | dot -Tpdf > servicios.pdf
```
produce un fichero PDF que muestra las relaciones de extensiones entre los servicios.

**shepherd-graph**
Emite en formato Dot/Graphviz por la salida estándar el grafo de dependencias de los servicios shepherd del sistema operativo definido en `fichero`. Véase Sección 8.17.4 [Servicios de Shepherd], página 420, para más información y un grafo de ejemplo.

### 8.15 Invocación de `guix deploy`

Ya hemos visto como usar declaraciones `operating-system` para gestionar la configuración de una máquina de manera local. Supongamos no obstante que necesita configurar múltiples máquinas—quizá esté gestionando un servicio en la web que se componga de varios servidores. `guix deploy` le permite usar las mismas declaraciones `operating-system` para gestionar múltiples máquinas remotas como un único “despliegue” lógico.

**Nota:** La funcionalidad descrita en esta sección está todavía en desarrollo y sujeta a cambios. Puede ponerse en contacto con nosotros a través de `guix-devel@gnu.org`.

`guix deploy fichero`

Dicha invocación llevará a cabo en las máquinas el despliegue al cual el `fichero` evalúe. Como ejemplo, `fichero` puede contener una definición como esta:

```plaintext
;; Este es un despliegue de Guix con una configuración en
;; mínima ("en los huesos"), sin servidor gráfico X11,
;; en una máquina con un daemon SSH escuchando en
;; localhost:2222. Una configuración como esta puede ser
;; apropiada para máquinas virtuales con puertos redirigidos
;; a la interfaz local de la máquina anfitriona.
(use-service-modules networking ssh)
(use-package-modules bootloaders)
(define %sistema
  (operating-system
    (host-name "despliegue-gnu")
    (timezone "Etc/UTC")
    (bootloader (bootloader-configuration
                  (bootloader grub-bootloader)
                  (target "/dev/vda")
                  (terminal-outputs '(console))))
    (file-systems (cons (file-system
                          (mount-point "/")
                          (device "/dev/vda1")
```

```
El fichero debe evaluar a una lista de objetos `machine`. Este ejemplo, durante el despliegue, creará una nueva generación en el sistema remoto que implemente la declaración `operating-system %sistema`. `environment` y `configuration` especifican cómo debe aprovisionarse la máquina—es decir, cómo se crean y gestionan los recursos computacionales. El ejemplo previo no crea ningún recurso, ya que `managed-host` es una máquina que ya está ejecutando el sistema Guix y está disponible a través de la red. Este es un caso particularmente simple; un despliegue más complejo puede implicar, por ejemplo, el arranque de máquinas virtuales a través de un proveedor de servidores privados virtuales (VPS). En cuyo caso se hubiera usado un tipo distinto en `environment`.

Tenga en cuenta que primero debe generar un par de claves en la máquina coordinadora para permitir al daemon exportar archivos firmados de ficheros en el almacén (véase Sección 4.11 [Invocación de guix archive], página 63):

```
# guix archive --generate-key
```

Cada máquina de destino debe autorizar a la clave de la máquina maestra para que acepte elementos del almacén que reciba de la coordinadora:

```
# guix archive --authorize < clave-publica-coordinadora.txt
```

`user`, in this example, specifies the name of the user account to log in as to perform the deployment. Its default value is `root`, but root login over SSH may be forbidden in some cases. To work around this, `guix deploy` can log in as an unprivileged user and employ `sudo` to escalate privileges. This will only work if `sudo` is currently installed on the remote and can be invoked non-interactively as `user`. That is, the line in `sudoers` granting `user` the ability to use `sudo` must contain the `NOPASSWD` tag. This can be accomplished with the following operating system configuration snippet:

```
(use-modules ...
  (gnu system)) ;for %sudoers-specification
```
(define %user "username")

(operating-system)
...
(sudoers-file
 (plain-file "sudoers"
  (string-append (plain-file-content %sudoers-specification)
   (format #f "~a ALL = NOPASSWD: ALL~%
    %user)))))

For more information regarding the format of the sudoers file, consult man sudoers.

**machine**

[Tipo de datos]
Tipo de datos que representa una máquina individual en un despliegue heterogéneo de Guix.

**operating-system**

El objeto de la configuración de sistema operativo a desplegar.

**environment**

Un objeto environment-type que describe cómo debe aprovisionarse la máquina.

**configuration** (predeterminado:

Un objeto que describe la configuración para el entorno (environment) de la máquina. Si environment tiene una configuración predeterminada, puede usarse #f. No obstante, si se usa #f para un entorno sin configuración predeterminada se emitirá un error.

**machine-ssh-configuration**

[Tipo de datos]
Tipo de datos que representa los parámetros del cliente SSH para una máquina con un entorno (environment) de tipo gestionado (managed-host-environment-type).

**host-name**

**build-locally?** (predeterminado: 

Si es falso, las derivaciones del sistema se construirán en la máquina sobre la que se realiza el despliegue.

**system**

El tipo de sistema que describe la arquitectura de la máquina sobre la que se realiza el despliegue—por ejemplo, "x86_64-linux".

**authorize?** (predeterminado: 

Si es verdadero, la clave de firma de la máquina coordinadora debe añadirse al anillo de claves del control de acceso (ACL) de la máquina remota.

**port** (predeterminado: 22)

**user** (predeterminada: "root")

**identity** (predeterminada: 

Cuando se especifica, indica la ruta al fichero que contiene la clave privada de SSH para la identificación con la máquina remota.
host-key (predeterminada: #f)
   Esta debería ser la clave SSH de la máquina, que puede ser más o menos así:

   ssh-ed25519 AAAAC3Nz... root@example.org

   Cuando host-key es #f, el servidor se identifica con el fichero
   ~/.ssh/known_hosts, igual que hace el cliente ssh de OpenSSH.

digital-ocean-configuration [Tipo de datos]
   Tipo de datos que representa el Droplet que debe crearse para la máquina con
   environment digital-ocean-environment-type.

ssh-key La ruta al fichero que contiene la clave privada de SSH usada para la
   identificación con la máquina remota. En el futuro este campo puede
   desaparecer.

tags Una lista de cadenas de etiquetas (“tags”) que identifican de manera
   unívoca a la máquina. Debe comprobarse que en un despliegue no existan
   dos máquinas que tengan el mismo conjunto de etiquetas.

region Descriptor (slug) de región de Digital Ocean, como "nyc3".

size Descriptor (slug) de tamaño de Digital Ocean, como "s-1vcpu-1gb"

enable-ipv6?
  Determina si droplet debe crearse con capacidad de usar redes IPv6 o no.

8.16 Ejecución de Guix en una máquina virtual

Para ejecutar Guix en una máquina virtual (VM), se puede usar tanto
la imagen de máquina virtual de Guix preconstruida que se distribuye en
'https://ftp.gnu.org/gnu/guix/guix-system-vm-image-1.0.1.x86_64-linux.xz'.
Esta imagen es una imagen comprimida con formato QCOW. Primero tendrá que
descomprimirla con xz -d y, una vez hecho, podrá proporcionarsela a un emulador como
QEMU (véase más detalles a continuación).

Esta imagen arranca en el entorno gráfico Xfce y contiene algunas herramientas usadas comúnmente. Puede instalar más software en la imagen mediante la ejecución de guix
command en un terminal (véase Sección 4.2 [Invocación de guix package], página 36). También puede reconfigurar el sistema en base a su fichero de configuración inicial, disponible como
/etc/config.scm (véase Sección 8.1 [Uso de la configuración del sistema], página 161).

En vez de usar esta imagen preconstruida, se puede construir una imagen propia para
máquina virtual mediante el uso de guix system vm-image (véase Sección 8.14 [Invocación de
guix system], página 399). La imagen devuelta se encuentra en formato qcow2, el cual el
emulador QEMU (https://qemu.org/) puede usar de manera eficiente.

Si ha construido su propia imagen, debe copiarla fuera del almacén y proporcionarse a sí
mismo permisos de escritura sobre dicha copia antes de usarla. En la invocación de QEMU
debe elegir un emulador de sistema que sea adecuado para su plataforma hardware. Ésta es
una invocación de QEMU mínima que arrancará el resultado de guix system vm-image en
hardware x86_64:

   $ qemu-system-x86_64 \
Aquí está el significado de cada una de esas opciones:

```
qemu-system-x86_64
```

Esto especifica la plataforma hardware a emular. Debe corresponder con el anfitrión.

```
-nic user,model=virtio-net-pci
```

Activa la pila de red en espacio de usuaria sin privilegios. El SO anfitrión puede acceder a la máquina virtualizada pero no al revés. Este es el modo más simple de poner la máquina en red. `model` especifica que dispositivo de red emular: `virtio-net-pci` es un dispositivo especial para sistemas operativos virtualizados y recomendado para la mayor parte de usos. Asumiendo que su plataforma de hardware es x86_64, puede obtener una lista de adaptadores de red disponibles ejecutando `qemu-system-x86_64 -nic model=help`.

```
-enable-kvm
```

Si su sistema tiene extensiones de virtualización por hardware, la activación de la implementación de máquinas virtuales (KVM) del núcleo Linux hará que la ejecución sea más rápida.

```
-m 1024
```

RAM disponible para el sistema operativo virtualizado, en mebibytes. El valor predeterminado es 128 MiB, que puede ser insuficiente para algunas operaciones.

```
-device virtio-blk,drive=midisco
```

Crea un dispositivo `virtio-blk` llamado “midisco”. `virtio-blk` es un mecanismo de “paravirtualización” de dispositivos de bloques que permite a QEMU obtener un mejor rendimiento que se emulase una unidad de disco completa. Véase la documentación de QEMU y KVM para más información.

```
-drive if=none,file=/tmp/imagen-qemu,id=midisco
```

Usa nuestra imagen QCOW, el fichero `/tmp/imagen-qemu`, como almacenamiento físico para la unidad “midisco”.

El guión `run-vm.sh` predeterminado que devuelve la invocación de `guix system vm` no añade una opción `-nic user` de manera predeterminada. Para obtener acceso a la red desde la máquina virtual añadida el servicio (dhcp-client-service) a su definición de sistema y arranque la máquina virtual mediante el uso de ‘`guix system vm config.scm` -nic user’. Un punto importante a tener en cuenta del uso de `-ni user` para la obtención de red es que `ping` no funcionará, puesto que usa el protocolo ICMP. Deberá usar una orden diferente para comprobar la conectividad a la red, por ejemplo `guix download`.

**8.16.1 Conexión a través de SSH**

Para activar SSH dentro de una máquina virtual debe añadir un servidor SSH como (openssh-service-type) en su máquina virtual (véase Sección 8.8.4 [Servicios de red], página 200). Además debe que redirigir el puerto SSH, el 22 por omisión, a la máquina anfitriona. Puede hacerlo con

```
'guix system vm config.scm' -nic user,model=virtio-net-pci,hostfwd=tcp::10022-:22
```
Para conectarse a la máquina virtual puede ejecutar

```bash
ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -p 10022
```

La `-p` indica a `ssh` el puerto al que se debe conectar. `-o UserKnownHostsFile=/dev/null` evita que `ssh` se queje cada vez que modifique su fichero `config.scm` y la orden `-o StrictHostKeyChecking=no` evita que tenga que autorizar la conexión a una máquina desconocida cada vez que se conecte.

### 8.16.2 Uso de `virt-viewer` con Spice

Como alternativa al cliente gráfico predeterminado de `qemu` puede usar `remote-viewer` del paquete `virt-viewer`. Para conectarse proporcione la opción `-spice port=5930,disable-ticketing` a `qemu`. Véase la sección previa para más información sobre cómo hacer esto.

Spice también le permite hacer cosas como compartir su portapapeles con su máquina virtual. Para activarlo debe proporcionar también las siguientes opciones a `qemu`:

```
-device virtio-serial-pci,id=virtio-serial0,max_ports=16,bus=pci.0,addr=0x5
-chardev spicevmc,name=vdagent,id=vdagent
-device virtserialport,nr=1,bus=virtio-serial0.0,chardev=vdagent,
    name=com.redhat.spice.0
```

También debe añadir el véase Sección 8.8.30 [Servicios misceláneos], página 384.

### 8.17 Definición de servicios

Las secciones anteriores muestran los servicios disponibles y cómo se pueden combinar en una declaración `operating-system`. ¿Pero cómo las definimos en primer lugar? ¿Y qué es un servicio en cualquier caso?

#### 8.17.1 Composición de servicios

Definimos un `servicio` como, de manera genérica, algo que extiende la funcionalidad del sistema operativo. Habitualmente un servicio es un proceso—un `daemon`—iniciado cuando el sistema arranca: un servidor de shell seguro, un servidor Web, el daemon de construcción de Guix, etc. A veces un servicio es un daemon cuya ejecución puede ser iniciada por otro daemon—por ejemplo, un servidor FTP iniciado por `inetd` o un servicio D-Bus activado por `dbus-daemon`. De manera ocasional, un servicio no se puede asociar a un daemon. Por ejemplo, el servicio “account” recopila cuentas de usuarias y se asegura que existen cuando el sistema se ejecuta; el servicio “udev” recopila reglas de gestión de dispositivos y los pone a disposición del daemon eudev; el servicio `/etc` genera el contenido del directorio `/etc` del sistema.

Los servicios de Guix se conectan a través de `extensiones`. Por ejemplo, el servicio de shell seguro `extiende` Shepherd—el sistema de inicio, el cual se ejecuta como PID 1—proveyendo las líneas de órdenes para arrancar y parar el daemon de shell seguro (véase Sección 8.8.4 [Servicios de red], página 200); el servicio UPower extiende el servicio D-Bus proporcionando su especificación `.service`, y extiende el servicio udev al que proporciona reglas de gestión de dispositivos (véase Sección 8.8.7 [Servicios de escritorio], página 237); el servicio del daemon de Guix extiende Shepherd proporcionando las líneas de órdenes para arrancar y parar el daemon, y extiende el servicio de cuentas proporcionando una lista de cuentas de usuarias de construcción que necesita (véase Sección 8.8.1 [Servicios base], página 183).
Al fin y al cabo, los servicios y sus relaciones de “extensión” forman un grafo acíclico dirigido (GAD). Si representamos los servicios como cajas y las extensiones como flechas, un sistema típico puede proporcionar algo de este estilo:

En la base, podemos ver el servicio del sistema, el cual produce el directorio que contiene todo lo necesario para ejecutar y arrancar el sistema, como es devuelto por la orden `guix system build`. Véase Sección 8.17.3 [Referencia de servicios], página 416, para aprender acerca de otros servicios mostrados aquí. Véase [system-extension-graph], página 406, para información sobre cómo generar esta representación para una definición particular de sistema operativo.

Técnicamente, las desarrolladoras pueden definir tipos de servicio para expresar estas relaciones. Puede haber cualquier número de servicios de un tipo dado en el sistema—por ejemplo, un sistema que ejecuta dos instancias del shell seguro GNU (lsh) tiene dos instancias de `lsh-service-type`, con parámetros diferentes.

La siguiente sección describe la interfaz programática para tipos de servicio y servicios.
8.17.2 Tipos de servicios y servicios

Un tipo de servicio es un nodo en el GAD descrito previamente. Empecemos con un ejemplo simple, el tipo de servicio para el daemon de construcción Guix (véase Sección 2.5 [Invocación de guix-daemon], página 15):

```
(define guix-service-type
  (service-type
    (name 'guix)
    (extensions
      (list (service-extension shepherd-root-service-type guix-shepherd-service)
            (service-extension account-service-type guix-accounts)
            (service-extension activation-service-type guix-activation)))
    (default-value (guix-configuration)))
```

Define tres cosas:
1. Un nombre, cuyo único propósito es facilitar la inspección y la depuración.
2. Una lista de extensiones de servicio, donde cada extensión designa el tipo de servicio a extender y un procedimiento que, dados los parámetros del servicio, devuelve una lista de objetos para extender el servicio de dicho tipo.
   Cada tipo de servicio tiene al menos una extensión de servicio. La única excepción es el tipo de servicio de arranque, que es el último servicio.
3. De manera opcional, un valor predeterminado para instancias de este tipo.

En este ejemplo, `guix-service-type` extiende tres servicios:

**shepherd-root-service-type**
El procedimiento `guix-shepherd-service` define cómo se extiende el servicio de Shepherd. Es decir, devuelve un objeto `<shepherd-service>` que define cómo se arranca y para `guix-daemon` (véase Sección 8.17.4 [Servicios de Shepherd], página 420).

**account-service-type**
`guix-accounts` crea la implementación de esta extensión para este servicio, la cual devuelve una lista de objetos `user-group` y `user-account` que representan las cuentas de usuarias de construcción (véase Sección 2.5 [Invocación de guix-daemon], página 15).

**activation-service-type**
Aquí `guix-activation` es un procedimiento que devuelve una expresión-G, que es un fragmento de código a ejecutar en “tiempo de activación”—por ejemplo, cuando el servicio se arranca.

Un servicio de este tipo se puede instanciar de esta manera:
```
(service guix-service-type
  (guix-configuration
    (build-accounts 5)
    (use-substitutes? #f)))
```

El segundo parámetro a la forma `service` es un valor que representa los parámetros de esta instancia específica del servicio. Véase [guix-configuration-type], página 191, para...
información acerca del tipo de datos guix-configuration. Cuando se omite el valor, se usa el valor predeterminado por guix-service-type:

(s service guix-service-type)

guix-service-type es bastante simple puesto que extiende otros servicios pero no es extensible a su vez.

El tipo de servicio para un servicio extendible puede tener esta forma:

(define udev-service-type
  (service-type (name 'udev)
    (extensions
      (list (service-extension shepherd-root-service-type udev-shepherd-service)))
    (compose concatenate) ; concatena la lista de reglas
    (extend (lambda (config rules)
      (match config
        (($ <udev-configuration> udev initial-rules)
          (udev-configuration
            (udev udev) ; el paquete udev usado
            (rules (append initial-rules rules)))))
        (else (rules))))))

Este es el tipo de servicio para el daemon de gestión de dispositivos eudev (https://wiki.gentoo.org/wiki/Project:Eudev). En comparación con el ejemplo previo, además de una extensión de shepherd-root-service-type, podemos ver dos nuevos campos:

compose Este es el procedimiento para componer la lista de extensiones en servicios de este tipo.
Los servicios pueden extender el servicio udev proporcionandole una lista de reglas; componemos estas extensiones mediante una simple concatenación.

extend Este procedimiento define cómo el valor del servicio se extiende con la composición de la extensión.
Las extensiones de udev se componen en una lista de reglas, pero el valor del servicio udev es en sí un registro <udev-configuration>. Por tanto aquí extenderemos el registro agregando la lista de reglas que contiene al final de la lista de reglas que se contribuyeron.

description Es una cadena que proporciona una descripción del tipo de servicio. Dicha cadena puede contener lenguaje de marcado Texinfo (véase Sección “Overview” en GNU Texinfo). La orden guix system search busca estas cadenas y las muestra (véase Sección 8.14 [Invocación de guix system], página 399).

Puede haber únicamente una instancia de un tipo de servicio extensible como udev-service-type. Si hubiese más, las especificaciones service-extension serían ambiguas.

¿Todavía aquí? La siguiente sección proporciona una referencia de la interfaz programática de los servicios.
8.17.3 Referencia de servicios

Ya hemos echado un vistazo a los tipos de servicio (véase Sección 8.17.2 [Tipos de servicios y servicios], página 414). Esta sección proporciona referencias sobre cómo manipular servicios y tipos de servicio. Esta interfaz se proporciona en el módulo (gnu services).

**service tipo [valor]** [Procedimiento Scheme]

- Devuelve un nuevo servicio de tipo, un objeto <service-type> (véase a continuación). valor puede ser cualquier objeto; representa los parámetros de esta instancia de servicio particular.
- Cuando se omite valor, se usa el valor predeterminado especificado por tipo; si type no especifica ningún valor, se produce un error.
- Por ejemplo, esto:

  (service openssh-service-type)

  es equivalente a esto:

  (service openssh-service-type
   (openssh-configuration))

  En ambos casos el resultado es una instancia de openssh-service-type con la configuración predeterminada.

**service? obj** [Procedimiento Scheme]

- Devuelve verdadero si obj es un servicio.

**service-kind servicio** [Procedimiento Scheme]

- Devuelve el tipo de servicio—es decir, un objeto <service-type>.

**service-value servicio** [Procedimiento Scheme]

- Devuelve el valor asociado con servicio. Representa sus parámetros.

Este es un ejemplo de creación y manipulación de un servicio:

```
(define s
  (service nginx-service-type
   (nginx-configuration
    (nginx nginx)
    (log-directory log-directory)
    (run-directory run-directory)
    (file config-file))))
```

```
(service? s)
⇒ #t

(eq? (service-kind s) nginx-service-type)
⇒ #t
```

La forma modify-services proporciona una manera fácil de cambiar los parámetros de algunos servicios de una lista como %base-services (véase Sección 8.8.1 [Servicios base], página 183). Evalúa a una lista de servicios. Por supuesto, siempre puede usar operaciones estándar sobre listas como map y fold para hacerlo (véase Sección “SRFI-1” en GNU Guile Reference Manual); modify-services proporciona simplemente una forma más concisa para este patrón común.
modify-services servicios (tipo variable => cuerpo) ... [Sintaxis Scheme]
Modifica los servicios listados en servicios de acuerdo a las cláusulas proporcionadas.
Cada cláusula tiene la forma:

(tipo variable => cuerpo)

donde *tipo* es un tipo de servicio—por ejemplo, guix-service-type—y *variable* es un identificador que se asocia dentro del *cuerpo* a los parámetros del servicio—por ejemplo, una instancia guix-configuration—del servicio original de dicho *ipo*.

El *cuerpo* debe evaluar a los nuevos parámetros del servicio, que serán usados para configurar el nuevo servicio. Este nuevo servicio reemplaza el original en la lista resultante. Debido a que los parámetros de servicio de un servicio se crean mediante el uso de define-record-type*, puede escribir un breve *cuerpo* que evalúe a los nuevos parámetros del servicio mediante el uso de la característica inherit que proporciona define-record-type* para heredar los valores antiguos.

Véase Sección 8.1 [Uso de la configuración del sistema], página 161, para ejemplos de uso.

A continuación se procede con la interfaz programática de los tipos de servicios. Es algo que debe conocer para escribir definiciones de nuevos servicios, pero no es cuando busque formas de personalizar su declaración operating-system.

**service-type** [Tipo de datos]
Esta es la representación de un tipo de servicio (véase Sección 8.17.2 [Tipos de servicios y servicios], página 414).

- **name**
  Es un símbolo, usado únicamente para simplificar la inspección y la depuración.

- **extensions**
  Una lista no vacía de objetos <service-extension> (véase a continuación).

- **compose** (predeterminado: #f)
  Si es #f, entonces el tipo de servicio denota servicios que no pueden extenderse—es decir, servicios que no pueden recibir "valores" de otros servicios.
  
  En otro caso, debe ser un procedimiento de un único parámetro. El procedimiento es invocado en fold-services y se le proporciona una lista de valores recibidos de las extensiones. Puede devolver un valor único.

- **extend** (predeterminado: #f)
  Si es #f, los servicios de este tipo no pueden extenderse.
  
  En otro caso, debe ser un procedimiento que acepte dos parámetros: fold-services lo invoca, proporcionándole el valor inicial del servicio como el primer parámetro y el resultado de aplicar compose a los valores de las extensiones como segundo parámetro. Debe devolver un valor que es un parámetro válido para la instancia del servicio.

Véase Sección 8.17.2 [Tipos de servicios y servicios], página 414, para ejemplos.
service-extension tipo-deseado calcula  [Procedimiento Scheme]
Devuelve una nueva extensión para servicios del tipo tipo-deseado. calcula debe ser un procedimiento de un único parámetro: es llamado en fold-services, proporcionandole el valor asociado con el servicio que proporcionala la extensión; debe devolver un valor válido para el servicio deseado.

service-extension? obj  [Procedimiento Scheme]
Devuelve verdadero si obj es una expresión-G.

De manera ocasional, puede desear simplemente extender un servicio existente. Esto implica la creación de un nuevo tipo de servicio y la especificación de la extensión deseada, lo cual puede ser engorroso; el procedimiento simple-service proporciona un atajo para ello.

simple-service nombre deseado valor  [Procedimiento Scheme]
Devuelve un servicio que extiende deseado con valor. Esto funciona creando una instancia única del tipo de servicio nombre, de la cual el servicio devuelto es una instancia.

Por ejemplo, esto extiende mcron (véase Sección 8.8.2 [Ejecución de tareas programadas], página 196) con una tarea adicional:

\[
\text{(simple-service 'mi-tarea-mcron mcron-service-type
\quad \#~(job '(next-hour (3)) "guix gc -F 2G"))}
\]

En el núcleo de la abstracción de los servicios se encuentra el procedimiento fold-services, que es responsable de la “compilación” de una lista de servicios en un único directorio que contiene todo lo necesario para arrancar y ejecutar el sistema—el directorio mostrado por la orden guix system build (véase Sección 8.14 [Invocación de guix system], página 399). En esencia, propaga las extensiones de servicios a través del grafo de servicios, actualizando los parámetros de cada nodo en el camino, hasta que alcanza el nodo raíz.

fold-services servicios [#:target-type
system-service-type]
Recorre servicios propagando sus extensiones hasta la raíz del tipo target-type; devuelve el servicio raíz tratado de la manera apropiada.

Por último, el módulo (gnu services) también define varios tipos esenciales de servicios, algunos de los cuales se enumeran a continuación.

system-service-type  [Variable Scheme]
Esta es la raíz del grafo de servicios. Produce el directorio del sistema como lo devuelve la orden guix system build.

boot-service-type  [Variable Scheme]
El tipo del “servicio de arranque”, que produce un guión de arranque. El guión de arranque es lo que ejecuta el disco inicial de RAM cuando se arranca.
Capítulo 8: Configuración del sistema

etc-service-type

El tipo del servicio /etc. Este servicio se usa para crear los ficheros en /etc y puede extenderse proporcionándole pares nombre/fichero como estas:

(list '("issue", (plain-file "issue" "¡Bienvenida!

En este ejemplo, el efecto sería la adición de un fichero /etc/issue que apunte al fichero proporcionado.

setuid-program-service-type

Tipo para el “servicio de programas setuid”. Este servicio recopila listas de nombres de ficheros ejecutables, proporcionados como expresiones-G, y los añade al conjunto de programas con setuid de root en el sistema (véase Sección 8.9 [Programas con setuid], página 389).

profile-service-type

Tipo del servicio que genera el perfil del sistema—es decir, los programas en /run/current-system/profile. Otros servicios pueden extenderlo proporcionándole listas de paquetes a añadir al perfil del sistema.

provenance-service-type

Es el tipo del servicio que registra los metadatos de procedencia en el sistema mismo. Crean varios ficheros en /run/current-system:

channels.scm

Es un “fichero de canales” que se le puede proporcionar a guix pull -C o guix time-machine -C, y que describe los canales usados para construir el sistema, si dicha información estaba disponible (véase Sección 4.7 [Canales], página 54).

configuration.scm

Este es el fichero que se proporciona como valor para el servicio provenance-service-type. De manera predeterminada, guix system reconfigure proporciona automáticamente el fichero de configuración del SO que recibió en la línea de órdenes.

provenance

Contiene la misma información que los otros dos ficheros, pero en un formato que se puede procesar más fácilmente.

En general, estas dos piezas de información (canales y el fichero de configuración) son suficientes para reproducir el sistema operativo “desde las fuentes”.

Advertencias: Esta información es necesaria para reconstruir su sistema operativo, pero no siempre es suficiente. En particular, configuration.scm en sí es insuficiente si no está autocontenido—si hace referencia a módulos externos de Guile o a ficheros adicionales. Si desea que configuration.scm sea autocontenido, le recomendamos que los módulos o ficheros a los que haga referencia sean parte de un canal. Además, la proveniencia de los metadatos es “silenciosa” en el sentido de que no cambia los bits que contiene su sistema, excepto por los bits de los metadatos en sí. Dos configuraciones de SO diferentes o conjuntos de canales pueden llevar al mismo sistema, bit a bit; cuando se usa
**provenance-service-type**, estos dos sistemas tendrán distintos metadatos y por lo tanto distintos nombres de fichero en el almacén, lo que hace no tan trivial dicha comparación.

Este servicio se añade automáticamente a la configuración de su sistema operativo cuando usa `guix system reconfigure`, `guix system init` o `guix deploy`.

### 8.17.4 Servicios de Shepherd

El módulo *(gnu services shepherd)* proporciona una forma de definir servicios gestionados por GNU Shepherd, que es el sistema de inicio—el primer proceso que se inicia cuando el sistema arranca, también conocido como PID 1 (véase Sección “Introducción” en *The GNU Shepherd Manual*).

Los servicios en Shepherd pueden depender de otros servicios. Por ejemplo, el daemon SSH puede tener que arrancarse tras el arranque del daemon syslog, lo cual a su vez puede suceder únicamente tras el montaje de todos los sistemas de ficheros. El sistema operativo simple definido previamente (véase Sección 8.1 [Uso de la configuración del sistema], página 161) genera un grafo de servicios como este:

![Grafo de servicios de Shepherd](image)
En realidad puede generar dicho grafo para cualquier definición de sistema operativo mediante el uso de la orden `guix system shepherd-graph` (véase [system-shepherd-graph], página 407).

`%shepherd-root-service` es un objeto de servicio que representa el PID 1, del tipo `shepherd-root-service-type`; puede ser extendido proporcionandole listas de objetos `<shepherd-service>`.

**shepherd-service**

El tipo de datos que representa un servicio gestionado por Shepherd.

- **provision**
  
  Una lista de símbolos que indican lo que proporciona el servicio.
  
  Esto son nombres que pueden proporcionarse a `herd start`, `herd status` y órdenes similares (véase Sección “Invoking herd” en *The GNU Shepherd Manual*). Véase Sección “Slots of services” en *The GNU Shepherd Manual*, para más detalles.

- **requirement** (predeterminada: `'(())`)
  
  Lista de símbolos que indican los servicios Shepherd de los que este depende.

- **one-shot?** (predeterminado: `#f`)
  
  Si este servicio es *one-shot*. Los servicios “one-shot” finalizan inmediatamente después de que su acción `start` se complete. Véase Sección “Slots of services” en *The GNU Shepherd Manual*, para más información.

- **respawn?** (predeterminado: `#t`)
  
  Indica si se debe reiniciar el servicio cuando se para, por ejemplo cuando el proceso subyacente muere.

- **start**

- **stop** (predeterminado: `#~(const #f)`)  
  
  Los campos `start` y `stop` hacen referencia a las características de Shepherd de arranque y parada de procesos respectivamente (véase Sección “Service De- and Constructors” en *The GNU Shepherd Manual*). Se proporcionan como expresiones-G que se expandirán en el fichero de configuración de Shepherd (véase Sección 6.7 [Expresiones-G], página 108).

- **actions** (predeterminadas: `'(())`)
  
  Esta es la lista de objetos `shepherd-action` (véase a continuación) que definen las acciones permitidas por el servicio, además de las acciones estándar `start` y `stop`. Las acciones que se listan aquí estarán disponibles como ordenes de `herd`:
  
  ```
  herd acción servicio [parámetros...]
  ```

- **auto-start?** (predeterminado: `#t`)
  
  Determina si Shepherd debe iniciar este servicio de manera automática. Si es `#f` el servicio debe iniciarse manualmente con `herd start`.

**documentación**

Una cadena de documentación, que se mostrará al ejecutar:

```
herd doc nombre-del-servicio
```
donde *nombre-del-servicio* es uno de los símbolos en *provision* (véase Sección “Invoking herd” en *The GNU Shepherd Manual*).

**modules** (predeterminados: %default-modules)

Esta es la lista de módulos que deben estar dentro del ámbito cuando *start* y *stop* son evaluados.

**shepherd-action**

[Tipo de datos]

Este es el tipo de datos que define acciones adicionales implementadas por un servicio Shepherd (vea previamente).

- **name** Símbolo que nombra la acción.
- **documentation** Esta es una cadena de documentación para la acción. Puede verse ejecutando:
  
  ```
  herd doc servicio action acción
  ```
- **procedure** Debe ser una expresión-G que evalúa a un procedimiento de al menos un parámetro, el cual es el “valor de ejecución” del servicio (véase Sección “Slots of services” en *The GNU Shepherd Manual*).

El siguiente ejemplo define una acción llamada *di-hola* que saluda amablemente a la usuaria:

```lisp
(shepherd-action
  (name 'di-hola)
  (documentation "¡Di hola!")
  (procedure #~(lambda (running . args)
         (format #t "¡Hola, compa! parámetros: ~s\n" args)
      #t)))
```

Asumiendo que esta acción se añade al servicio *ejemplo*, puede ejecutar:

```bash
# herd di-hola ejemplo
¡Hola, compa! parámetros: ()
# herd di-hola ejemplo a b c
¡Hola, compa! parámetros: ("a" "b" "c")
```


**shepherd-root-service-type**

[Variable Scheme]

El tipo de servicio para el “servicio raíz” de Shepherd—es decir, PID 1.

El tipo de servicio que las extensiones declaran cuando desean crear servicios shepherd (véase Sección 8.17.2 [Tipos de servicios y servicios], página 414, para un ejemplo). Cada extensión debe pasar una lista de <shepherd-service>.

**%shepherd-root-service**

[Variable Scheme]

Este servicio representa el PID 1.
9 Documentación

En la mayor parte de casos, los paquetes instalados con Guix contienen documentación. Hay dos formatos principales de documentación: “Info”, un formato hipertextual navegable usado para software GNU, y “páginas de manual” (o “páginas man”), la documentación lineal encontrada tradicionalmente en Unix. Se accede a los manuales Info con la orden `info` o con Emacs, y las páginas man con `man`.

Puede buscar documentación de software instalado en su sistema por palabras clave. Por ejemplo, la siguiente orden busca información sobre “TLS” en manuales Info:

```
$ info -k TLS
"(emacs)Network Security" -- STARTTLS
"(emacs)Network Security" -- TLS
"(gnutls)Core TLS API" -- gnutls_certificate_set_verify_flags
"(gnutls)Core TLS API" -- gnutls_certificate_set_verify_function
...
```

La orden siguiente busca por la misma palabra clave en páginas man:

```
$ man -k TLS
SSL (7) - OpenSSL SSL/TLS library
certtool (1) - GnuTLS certificate tool
...
```

Estas búsquedas son completamente locales en su máquina de modo que tiene la garantía de que la documentación que encuentra corresponde con lo que está realmente instalado, puede acceder a ella sin conexión a la red, y se respeta su privacidad.

Una vez tenga estos resultados, puede ver la documentación relevante mediante la ejecución de, digamos:

```
$ info "(gnutls)Core TLS API"
```

o:

```
$ man certtool
```

10 Instalación de ficheros de depuración

Los programas binarios, como los producidos por los compiladores GCC por ejemplo, se escriben típicamente en el formato ELF, con una sección que contiene información de depuración. La información de depuración es lo que permite que el depurador, GDB, asocie código binario a código fuente; es necesaria para depurar un programa compilado en condiciones adecuadas.

El problema con la información de depuración es que ocupa un espacio considerable en el disco. Por ejemplo, la información de depuración de la biblioteca C de GNU ocupa más de 60 MiB. Por tanto, como usuaria, mantener toda la información de depuración de todos los programas instalados no es habitualmente una opción. No obstante, el ahorro de espacio no debe ser impedir la depuración—especialmente en el sistema GNU, que debería facilitar a sus usuarias ejercitar su libertad de computación (véase Sección 1.2 [Distribución GNU], página 3).

Afortunadamente, las utilidades binarias GNU (Binutils) y GDB proporcionan un mecanismo que permite a las usuarias obtener lo mejor de ambos mundos: la información de depuración puede extraerse de los binarios y almacenarse en ficheros separados. GDB es capaz entonces de cargar la información de depuración desde esos ficheros, cuando estén disponibles (véase Sección “Separate Debug Files” en Debugging with GDB).

La distribución GNU toma ventaja de este hecho almacenando la información de depuración en el subdirectorio lib/debug de una salida separada del paquete llamada debug (véase Sección 4.4 [Paquetes con múltiples salidas], página 47). Las usuarias pueden elegir si instalan la salida debug de un paquete cuando la necesitan. Por ejemplo, la siguiente orden instala la información de depuración para la biblioteca C de GNU y para GNU Guile.

    guix install glibc:debug guile:debug

Se debe decir entonces a GDB que busque los ficheros de depuración en el perfil de la usuaria, proporcionando un valor a la variable debug-file-directory (considere hacerlo en el fichero ~/.gdbinit, véase Sección “Startup” en Debugging with GDB):

    (gdb) set debug-file-directory ~/.guix-profile/lib/debug

A partir de ese momento GDB obtendrá la información de depuración de los ficheros .debug bajo ~/.guix-profile/lib/debug.

Además, probablemente desee que GDB sea capaz de mostrar el código fuente que está depurando. Para hacerlo, tiene que desempaquetar el código fuente del paquete de su interés (obtenido con guix build --source, véase Sección 7.1 [Invocación de guix build], página 119) e indicar a GDB que el directorio de fuentes mediante el uso de la orden directory (véase Sección “Source Path” en Debugging with GDB).

El mecanismo de la salida debug en Guix se implementa por el sistema de construcción gnu-build-system (véase Sección 6.3 [Sistemas de construcción], página 85). Ahora mismo necesita una activación explícita—la información de depuración está disponible únicamente para paquetes con definiciones que declaren explícitamente una salida debug. Esto puede cambiarse por una activación implícita en el futuro si nuestras granjas de construcción pueden soportar la carga. Para comprobar si un paquete tiene una salida debug, use guix package --list-available (véase Sección 4.2 [Invocación de guix package], página 36).
11 Actualizaciones de seguridad

De manera ocasional, vulnerabilidades importantes de seguridad se descubren en los paquetes de software y deben aplicarse parches. Las desarrolladoras de Guix tratan de seguir las vulnerabilidades conocidas y aplicar parches tan pronto como sea posible en la rama master de Guix (todavía no proporcionamos una rama “estable” que contenga únicamente actualizaciones de seguridad). La herramienta guix lint ayuda a las desarrolladoras a encontrar versiones vulnerables de paquetes de software en la distribución:

```bash
$ guix lint -c cve
gnu/packages/gcc.scm:334:2: gcc@4.9.3: probablemente vulnerable a CVE-2015-5276
gnu/packages/image.scm:312:2: openjpeg@2.1.0: probablemente vulnerable a CVE-2016-1923, CVE-2016-1924
...
```

Véase Sección 7.7 [Invocación de guix lint], página 141, para más información.

Guix sigue una disciplina funcional de gestión de paquetes (véase Capítulo 1 [Introducción], página 2), lo que implica que, cuando se cambia un paquete, todos los paquetes que dependen de él deben ser reconstruidos. Esto puede ralentizar de manera significativa el despliegue de correcciones en paquetes básicos como libc o Bash, ya que básicamente la distribución al completo debe reconstruirse. El uso de binarios preconstruidos ayuda (véase Sección 4.3 [Sustituciones], página 44), pero el despliegue aún puede tomar más tiempo del deseado.

Para afrontar esto, Guix implementa injertos, un mecanismo que permite un rápido despliegue de actualizaciones críticas sin los costes asociados con una reconstrucción completa de la distribución. La idea es reconstruir únicamente el paquete que hace falta parchear, y entonces “injertarlo” en los paquetes explícitamente instalados por la usuaria y que previamente hacían referencia al paquete original. El coste de realizar un injerto es menor que una reconstrucción completa de la cadena de dependencias.

Por ejemplo, supongamos que es necesario incorporar una actualización de seguridad en Bash. Las desarrolladoras de Guix proporcionarán una definición de paquete para la versión “corregida” de Bash, digamos bash-fixed, de la manera habitual (véase Sección 6.2 [Definición de paquetes], página 77). Una vez hecho, la definición original del paquete es aumentada con un campo replacement que apunta al paquete que contiene la corrección del error:

```scheme
(define bash
  (package
    (name "bash")
    ...
    (replacement bash-fixed)))
```

De ahí en adelante, cualquier paquete que dependa directa o indirectamente de Bash—como informa de ello guix gc --requisites (véase Sección 4.5 [Invocación de guix gc], página 48)—que se instale se “reescribe” automáticamente para hacer referencia a bash-fixed en vez de bash. Este proceso de injerto toma un tiempo proporcional al tamaño del paquete, normalmente menos de un minuto para un paquete “medio” en una máquina reciente. El injertado es recursivo: cuando una dependencia indirecta requiere un injerto, el injerto se “propagará” hasta el paquete que la usuaria esté instalando.
Actualmente, la longitud del nombre y la versión del injerto y aquella del paquete que reemplaza (bash-fixed y bash en el ejemplo previo) debe ser igual. Esta restricción viene principalmente del hecho de que el injertado funciona mediante la aplicación de parches en ficheros, incluyendo ficheros binarios, directamente. Otras restricciones pueden ser aplicables: por ejemplo, durante la adición de un injerto a un paquete que proporciona una biblioteca compartida, la biblioteca compartida y su reemplazo deben tener el mismo SONAME y deben ser compatibles a nivel binario.

La opción de línea de órdenes --no-grafts le permite anular voluntariamente el proceso de injerto (véase Sección 7.1.1 [Opciones comunes de construcción], página 119). Por tanto, la orden:

```bash
guix build bash --no-grafts
```

devuelve el nombre de fichero del almacén de la versión original de Bash, mientras que:

```bash
guix build bash
```

devuelve el nombre de fichero del almacén de la versión “corregida”, reemplazo de Bash. Esto le permite distinguir entre las dos variantes de Bash.

Para verificar a qué Bash hace referencia su perfil al completo, puede ejecutar (véase Sección 4.5 [Invocación de guix gc], página 48):

```bash
guix gc -R 'readlink -f ~/.guix-profile' | grep bash
```

... y compare los nombres de fichero del almacén que obtendrá con los ejemplos previos. Del mismo modo, para una generación completa del sistema Guix:

```bash
guix gc -R 'guix system build mi-configuración.scm' | grep bash
```

Por último, para comprobar qué versión de Bash están usando los procesos en ejecución, puede usar la orden lsof:

```bash
lsof | grep /gnu/store/.*bash
```
12 Lanzamiento inicial

El lanzamiento inicial en nuestro contexto hace referencia a cómo la distribución se construye “de la nada”. Recuerde que el entorno de construcción de una derivación no contiene más que sus entradas declaradas (véase Capítulo 1 [Introducción], página 2). Por lo que hay un evidente problema “del huevo y la gallina”: ¿cómo se construye el primer paquete? ¿Cómo se compila el primer compilador? Fíjese que esta es una cuestión de interés únicamente para la hacker curiosa, no para la usuaria normal, así que puede pasar por encima está sección sin ninguna vergüenza si se considera una “usuaria normal”.

El sistema GNU está compuesto principalmente de código C, con libc en su base. El sistema de construcción GNU en sí asume la disponibilidad del shell Bourne y las herramientas de línea de órdenes proporcionadas por GNU Coreutils, Awk, Findutils, ‘sed’ y ‘grep’. Además, los programas de construcción—programas que ejecutan ./configure, make, etc.—están escritos en Scheme Guile (véase Sección 6.5 [Derivaciones], página 101). Consecuentemente, para ser capaz de construir cualquier cosa, desde cero, Guix depende de binarios preconstruidos de Guile, GCC, Binutils, libc y otros paquetes mencionados anteriormente—los binarios del lanzamiento inicial.

Estos binarios del lanzamiento inicial se “dan por supuestos”, aunque se pueden volver a crear si se necesita (más sobre esto más adelante).

En i686-linux y x86_64-linux el proceso del lanzamiento inicial de Guix es más elaborado, véase Sección 12.1 [Lanzamiento inicial a partir de la semilla binaria reducida], página 427.

12.1 El lanzamiento inicial a partir de la semilla binaria reducida

Guix—al igual que otras distribuciones de GNU/Linux—se lanza inicialmente desde un conjunto de binarios de manera tradicional: un shell Bourne, herramientas de línea de órdenes que proporcionan GNU Coreutils, Awk, Findutils, ‘sed’ y ‘grep’ y Guile, GCC, Binutils y la biblioteca de C de GNU (véase Capítulo 12 [Lanzamiento inicial], página 427). Habitualmente dichos binarios se “dan por hecho”.

El dar por hecho estos binarios significa que consideramos que son una “semilla” correcta y fiable para la construcción del sistema completo. En esta asunción yace un problema: el tamaño combinado de dichos binarios necesarios para el lanzamiento inicial es cercano a 250MB (véase Sección “Bootstrappable Builds” en GNU Mes). El auditado o incluso la inspección de dichos binarios es prácticamente imposible.

En i686-linux y x86_64-linux, Guix basa su lanzamiento inicial en una “semilla binaria reducida”.

El lanzamiento inicial a partir de la semilla binaria reducida elimina las herramientas más críticas—desde la perspectiva de la confianza—de los binarios necesarios: GCC, Binutils y la biblioteca de C de GNU se sustituyen por: bootstrap-mescc-tools (un pequeño ensamblador y enlazador) y bootstrap-mes (un pequeño intérprete de Scheme y un compilador de C escrito en Scheme y la biblioteca de C de Mes, compilada para TinyCC y para

---

1 Nos gustaría decir: “lanzamiento inicial completamente desde las fuentes” pero, aunque estamos trabajando por alcanzar ese objetivo, sería una hipérbole usar dicho término para describir el proceso que seguimos en la actualidad.
GCC). Mediante el uso de estas nuevas semillas binarias y un nuevo conjunto de paquetes\(^2\) las herramientas que “faltan” (Binutils, GCC y la biblioteca de C de GNU) se construyen desde sus fuentes. Desde ese punto en adelante se continua el proceso de lanzamiento inicial tradicional. Esta aproximación ha reducido el tamaño de los binarios necesarios para el lanzamiento inicial cerca de 130MB. Se está trabajando en reducir más aún este tamaño. Si esto le interesa, puede únase en el canal \#bootstrappable de la red IRC de Freenode.

A continuación se encuentra el grafo de dependencias generado para gcc-mesboot, el compilador del lanzamiento inicial usado para construir el resto de GuixSD.

\(^2\) nyacc-boot, mes-boot, tcc-boot0, tcc-boot, make-mesboot0, diffutils-mesboot, binutils-mesboot0, gcc-core-mesboot, mesboot-headers, glibc-mesboot0, gcc-mesboot0, binutils-mesboot, make-mesboot, gcc-mesboot1, gcc-mesboot1-wrapper, glibc-headers-mesboot, glibc-mesboot, gcc-mesboot y gcc-mesboot-wrapper.
12.2 Preparación para usar los binarios del lanzamiento inicial

La figura previa muestra el auténtico inicio del grafo de dependencias de la distribución, correspondiente a las definiciones de paquete del módulo (gnu packages bootstrap). Un gráfico similar puede generarse con guix graph (véase Sección 7.9 [Invocación de guix graph], página 145), más o menos así:

```bash
guix graph -t derivation \
-e '(@@ (gnu packages bootstrap) %bootstrap-gcc)' \
| dot -Tps > gcc.ps
```

o, para la semilla binaria reducida del lanzamiento inicial

```bash
guix graph -t derivation \
-e '(@@ (gnu packages bootstrap) %bootstrap-mes)' \
| dot -Tps > mes.ps
```

En este nivel de detalle, las cosas son ligeramente complejas. Primero, Guile en sí consiste en un ejecutable ELF, junto a muchas fuentes y ficheros compilados Scheme que se cargan dinámicamente durante la ejecución. Esto se almacena en el archivador tar

guile-2.0.7.tar.xz mostrado en este grafo. Este archivador es parte de la distribución de “fuentes” de Guix, y se inserta en el almacén con add-to-store (véase Sección 6.4 [El almacén], página 98).

¿Pero cómo escribimos una derivación que extraiga este archivador y lo añada al almacén? Para resolver este problema, la derivación guile-bootstrap-2.0.drv—la primera en construirse—usa bash como su constructor, que ejecuta build-bootstrap-guile.sh, que a su vez llama a tar para extraer el archivador. Por tanto, bash, tar, xz y mkdir son binarios enlazados estáticamente, también parte de la distribución de fuentes de Guix, cuyo único propósito es permitir la extracción del archivador de Guile.

Una vez que guile-bootstrap-2.0.drv se ha construido, tenemos un Guile funcional que se puede usar para ejecutar los programas de construcción siguientes. Su primera tarea es descargar los archivadores que contienen los otros binarios preconstruidos—esto es lo que las derivaciones .tar.xz.drv hacen. Módulos Guix como ftp-client.scm se usan para este propósito. Las derivaciones module-import.drv importan esos módulos en un directorio del almacén, manteniendo la distribución de carpetas. Las derivaciones module-import-compiled.drv compilan esos módulos, y los escriben en un directorio con la distribución de carpetas correcta. Esto corresponde al parámetro #:modules de build-expression-derivation (véase Sección 6.5 [Derivaciones], página 101).

Finalmente, los archivadores tar son extraídos por las derivaciones gcc-bootstrap-0.drv, glibc-bootstrap-0.drv, or bootstrap-mes-0.drv y bootstrap-mescc-tools-0.drv, hasta el punto en el que disponemos de una cadena de herramientas C funcional.

**Construcción de las herramientas de construcción**

El lanzamiento inicial está completo cuando tenemos una cadena de herramientas completa que no depende en las herramientas preconstruidas del lanzamiento inicial descritas previamente. Este requisito de no-dependencia se verifica comprobando si los ficheros de la cadena de herramientas final contienen referencias a directorios de /gnu/store de las entradas del lanzamiento. El proceso que lleva a esta cadena de herramientas “final” es descrito por las definiciones de paquetes encontradas en el módulo (gnu packages commencement).

La orden guix graph nos permite “distanciarnos” en comparación con el grafo previo, mirando al nivel de objetos de paquetes en vez de derivaciones individuales—recuerde que un paquete puede traducirse en varias derivaciones, típicamente una derivación para descargar sus fuentes, una para construir los módulos Guile que necesita y uno para realmente construir el paquete de las fuentes. La orden:

```
        guix graph -t bag \
                -e '(@@ (gnu packages commencement) 
                    glibc-final-with-bootstrap-bash)'
                | dot -Tps > t.ps
```
produce el grafo de dependencias que lleva a la biblioteca C “final”\(^3\), mostrado a continuación.

<table>
<thead>
<tr>
<th>glibc-intermediate-2.24</th>
</tr>
</thead>
<tbody>
<tr>
<td>linux-libre-headers-4.4.18</td>
</tr>
<tr>
<td>lexinfo-6.3</td>
</tr>
<tr>
<td>gcc-cross-boot0-4.9.4</td>
</tr>
<tr>
<td>ld-wrapper-x86_64-guix-linux-gnu-0</td>
</tr>
<tr>
<td>perl-boot0-5.24.0</td>
</tr>
<tr>
<td>binutils-cross-boot0-2.27</td>
</tr>
<tr>
<td>findutils-boot0-4.6.0</td>
</tr>
<tr>
<td>file-boot0-5.28</td>
</tr>
<tr>
<td>diffutils-boot0-3.5</td>
</tr>
<tr>
<td>make-boot0-4.2.1</td>
</tr>
<tr>
<td>gcc-bootstrap-0</td>
</tr>
<tr>
<td>binutils-bootstrap-0</td>
</tr>
<tr>
<td>bootstrap-binaries-0</td>
</tr>
<tr>
<td>glibc-bootstrap-0</td>
</tr>
<tr>
<td>gcc-bootstrap-0</td>
</tr>
<tr>
<td>bootstrap-binaries-0</td>
</tr>
</tbody>
</table>

La primera herramienta que se construye con los binarios del lanzamiento inicial es GNU Make—marcado como `make-boot0` en el grafo—, que es un pre-requisito para todos los paquetes siguientes. Una vez hecho se construyen Findutils y Diffutils.

Después viene la primera fase de Binutils y GCC, construidas como herramientas pseudo-cruzadas—es decir, con `--target` igual a `--host`. Se usan para construir libc. Gracias a este truco de compilación cruzada, se garantiza que esta libc no tendrá ninguna referencia a la cadena de herramientas inicial.

Posteriormente se construyen las herramientas Binutils y GCC (no mostradas previamente) finales, y enlazan los programas contra la libc recién construida. Esta cadena de herramientas se usa para construir otros paquetes usados por Guix y el sistema de construcción GNU: Guile, Bash, Coreutils, etc.

¡Y voilà! En este punto tenemos un conjunto completo de herramientas de construcción esperadas por el sistema de construcción GNU. Están en la variable `%final-inputs` del

\(^3\) Puede haberse dado cuenta de la etiqueta `glibc-intermediate`, sugiriendo que no es completamente final, pero como es una buena aproximación, la consideraremos final.
módulo (gnu packages commencement), y se usan implícitamente por cualquier paquete que use gnu-build-system (véase Sección 6.3 [Sistemas de construcción], página 85).

**Construir los binarios de lanzamiento**

Debido a que la cadena de herramientas final no depende de los binarios de lanzamiento, estos rara vez necesitan ser actualizados. No obstante, es útil tener una forma automatizada de producirlos en caso de que se dé una actualización, y esto es lo que proporciona el módulo (gnu packages make-bootstrap).

La siguiente orden construye los archivadores que contienen los binarios de lanzamiento (Binutils, GCC, glibc para el lanzamiento inicial tradicional y linux-libre-headers, bootstrap-mescc-tools y bootstrap-mes para el lanzamiento inicial basado en la semilla binaria reducida, y Guile y un archivador que contiene una mezcla de Coreutils y otras herramientas básicas de línea de órdenes):

```
guix build bootstrap-tarballs
```

Los archivadores “tar” generados son aquellos a cuya referencia debe encontrarse en el módulo (gnu packages bootstrap) mencionado al inicio de esta sección.

¿Todavía aquí? Entonces quizás se habrá empezado a preguntar: ¿cuándo llegamos a un punto fijo? ¡Esa es una pregunta interesante! La respuesta es desconocida, pero si pudiese investigar más a fondo (y tiene unos recursos computacionales y de almacenamiento significativos para hacerlo) háganoslo saber.

**Reducción del conjunto de binarios de lanzamiento**

Nuestros binarios de lanzamiento actualmente incluyen GCC, GNU Libc, Guile, etc. ¡Eso es mucho código binario! ¿Por qué es eso un problema? Es un problema porque esos grandes fragmentos de código binario no son auditables en la práctica, lo que hace difícil establecer qué código fuente los produjo. Cada binario no-auditável también nos deja vulnerables a puertas traseras en los compiladores, como describió Ken Thompson en su publicación de 1984 Reflections on Trusting Trust.

Esto se mitiga por el hecho de que nuestros binarios de lanzamiento fueron generados por una revisión anterior de Guix. No obstante, esto no posee el nivel de transparencia que obtenemos en el resto del grado de dependencias de los paquetes, donde Guix siempre nos da una asociación de fuente-a-binario. Por lo tanto, nuestro objetivo es reducir el conjunto de binarios de lanzamiento al mínimo posible.

The Bootstrappable.org web site (https://bootstrappable.org) lists on-going projects to do that. One of these is about replacing the bootstrap GCC with a sequence of assemblers, interpreters, and compilers of increasing complexity, which could be built from source starting from a simple and auditable assembler.

Nuestro primer logro de importancia es la sustitución de GCC, la biblioteca de C de GNU y Binutils por MesCC-Tools (un enlazador hexadecimal y un macro-ensamblador) y Mes (véase GNU Mes, un intérprete de Scheme y compilador de C en Scheme). Ni MesCC-Tools ni Mes pueden lanzarse inicialmente a sí mismas completamente todavía y por lo tanto se inyectan como semillas binarias. A esto es lo que llamamos la semilla binaria reducida del lanzamiento inicial, ¡ya que ha reducido a la mitad el tamaño de nuestros binarios del lanzamiento inicial! También ha eliminado el binario del compilador de C; los paquetes de
Guix en i686-linux y x86_64-linux se generan a partir de un lanzamiento inicial sin ningún binario que sea un compilador de C.

Se está trabajando en hacer que MesCC-Tools y Mes puedan lanzarse inicialmente de manera completa, y también se buscan otros binarios para el lanzamiento inicial. ¡Su ayuda es bienvenida!
13 Transportar a una nueva plataforma

Como se explicó previamente, la distribución GNU es autocontenida, lo cual se consigue dependiendo de unos “binarios del lanzamiento inicial” preconstruidos (véase Capítulo 12 [Lanzamiento inicial], página 427). Estos binarios son específicos para un núcleo del sistema operativo, arquitectura de la CPU e interfaz binaria de aplicaciones (ABI). Por tanto, para transportar la distribución a una nueva plataforma que no está soportada todavía, se deben construir estos binarios del lanzamiento inicial, y actualizar el módulo (gnu packages bootstrap) para usarlos en dicha plataforma.

Por suerte, Guix puede compilar de forma cruzada esos binarios del lanzamiento inicial. Cuando todo va bien, y asumiendo que la cadena de herramientas GNU soporta para la plataforma deseada, esto puede ser tan simple como ejecutar una orden así:

```
    guix build --target=armv5tel-linux-gnueabi bootstrap-tarballs
```

Para que esto funcione, el procedimiento glibc-dynamic-linker en (gnu packages bootstrap) debe aumentarse para devolver el nombre de fichero correcto para el enlazador dinámico de libc en dicha plataforma; de igual manera, system->linux-architecture en (gnu packages linux) debe modificarse para la nueva plataforma.

Una vez construidos, el módulo (gnu packages bootstrap) debe ser actualizado para hacer referencia a estos binarios en la plataforma deseada. Esto es, los hash y las URL de los archivadores del lanzamiento inicial de la nueva plataforma deben añadirse junto a aquellos de las plataformas disponibles actualmente. El archivador tar del Guile usado para el lanzamiento inicial se trata de forma especial: se espera que esté disponible localmente, y gnu/local.mk tiene reglas que lo descargan para las arquitecturas disponibles; se debe añadir una regla para la nueva plataforma también.

En la práctica puede haber algunas complicaciones. Primero, puede ser que la tripleta extendida GNU que especifica un ABI (como el sufijo eabi previamente) no es reconocida por todas las herramientas GNU. Típicamente, glibc reconoce algunas de ellas, mientras que GCC usa una opción de configuración extra --with-abi (vea gcc.scm para ejemplos de como manejar este caso). En segundo lugar, algunos de los paquetes necesarios pueden fallar en su construcción para dicha plataforma. Por último, los binarios generados pueden estar defectuosos por alguna razón.
14 Contribuir

Este proyecto es un esfuerzo colaborativo, y ¡necesitamos su ayuda para que crezca! Por favor, contacte con nosotros en guix-devel@gnu.org y en #guix en la red IRC Freenode. Estamos abiertas a ideas, informes de errores, parches y cualquier cosa que pueda ser de ayuda para el proyecto. Especialmente se agradece ayuda en empaquetamiento (véase Sección 14.4 [Guías de empaquetamiento], página 438).

We want to provide a warm, friendly, and harassment-free environment, so that anyone can contribute to the best of their abilities. To this end our project uses a “Contributor Covenant”, which was adapted from https://contributor-covenant.org/. You can find a local version in the CODE-OF-CONDUCT file in the source tree.

Las contribuidoras no están obligadas a usar su nombre legal en los parches ni en la comunicación on-line; pueden usar cualquier nombre o seudónimo de su elección.

14.1 Construcción desde Git

Si quiere picar en el mismo Guix se recomienda usar la última versión del repositorio Git:

```
git clone https://git.savannah.gnu.org/git/guix.git
```

¿Cómo puede asegurarse de haber obtenido una copia genuina del repositorio? Guix proporciona una herramienta `identify` su copia, pero primero debe comprobar que esta herramienta es genuina para iniciar la cadena de confianza. Para hacerlo, ejecute:

```
git verify-commit 'git log --format=%H build-aux/git-authenticate.scm'
```

La salida debe ser parecida a esta:

```
gpg: Firmado el vie 27 dic 2019 01:27:41 CET
    usando RSA clave 3CE464558A84FDC69DB40CFB090B11993D9AEBB5
...  
gpg: Firmado el vie 27 dic 2019 01:25:22 CET
    usando RSA clave 3CE464558A84FDC69DB40CFB090B11993D9AEBB5
...  
```

... que significa que los cambios en este fichero están todos firmados con la clave 3CE464558A84FDC69DB40CFB090B11993D9AEBB5 (puede tener que obtener la clave de un servidor de claves, si no lo ha hecho todavía).

De aquí en adelante, puede identificar todos las revisiones incluidas en su copia ejecutando:

```
made authenticate
```

La primera ejecución puede tardar varios minutos, pero las ejecuciones siguientes son más rápidas.

**Nota:** Se recomienda que ejecute `make authenticate` tras cada ejecución de `git pull`. Así se asegura de que está recibiendo cambios válidos del repositorio.

El modo más fácil de preparar un entorno de desarrollo para Guix es, por supuesto, ¡usando Guix! Las siguientes órdenes inician un nuevo intérprete donde todas las dependencias y las variables de entorno apropiadas están listas para picar código en Guix:

```
guix environment guix --pure
```
Véase Sección 5.1 [Invocación de guix environment], página 66, para más información sobre dicha orden.

Si no puede usar Guix para construir Guix desde una copia de trabajo, son necesarios los paquetes siguientes además de los mencionados en las instrucciones de instalación (véase Sección 2.2 [Requisitos], página 7).

- GNU Autoconf (https://gnu.org/software/autoconf/);
- GNU Automake (https://gnu.org/software/automake/);
- GNU Gettext (https://gnu.org/software/gettext/);
- GNU Texinfo (https://gnu.org/software/texinfo/);
- Graphviz (https://www.graphviz.org/);
- GNU Help2man (optional) (https://www.gnu.org/software/help2man/).

En Guix se pueden añadir las dependencias opcionales ejecutando guix environment con --ad-hoc:

```
  guix environment guix --pure --ad-hoc help2man git strace
```

Ejecute ./bootstrap para generar la infraestructura del sistema de construcción usando Autoconf y Automake. Si obtiene un error como este:

```
configure.ac:46: error: possibly undefined macro: PKG_CHECK_MODULES
```

probablemente significa que Autoconf no pudo encontrar el fichero pkg.m4, que proporciona pkg-config. Asegúrese de que pkg.m4 está disponible. Lo mismo aplica para el conjunto de macros guile.m4 que proporciona Guile. Por ejemplo, si ha instalado Automake en /usr/local, no va a buscar ficheros .m4 en /usr/share. En ese caso tiene que ejecutar la siguiente orden:

```
  export ACLOCAL_PATH=/usr/share/aclocal
```

Véase Sección “Macro Search Path” en The GNU Automake Manual para más información.

Una vez terminada, ejecute ./configure como siempre. Asegúrese de pasar --localstatedir=directorio, donde directorio es el valor de localstatedir usado en su instalación actual (véase Sección 6.4 [El almacén], página 98, para información sobre esto). Recomendamos usar el valor /var

Finalmente, tiene que ejecutar make check para iniciar las pruebas (véase Sección 2.3 [Ejecución de la batería de pruebas], página 8). Si algo falla, eche un vistazo a las instrucciones de instalación (véase Capítulo 2 [Instalación], página 5) o envíe un mensaje—en Inglés—a la lista de correo.

### 14.2 Ejecución de Guix antes de estar instalado

Para mantener un entorno de trabajo estable, encontrará útil probar los cambios hechos en su copia de trabajo local sin instalarlos realmente. De esa manera, puede distinguir entre su sombrero de “usuaria final” y el traje de “harapos”.

Para dicho fin, todas las herramientas de línea de órdenes pueden ser usadas incluso si no ha ejecutado make install. Para hacerlo, primero necesita tener un entorno con todas las
Capítulo 14: Contribuir

dependencias disponibles (véase Sección 14.1 [Construcción desde Git], página 435), y entonces añada al inicio de cada orden ./pre-inst-env (el guión pre-inst-env se encuentra en la raíz del árbol de compilación de Guix; se genera con ./configure). Un ejemplo:\footnote{La opción \texttt{-E} a \texttt{sudo} asegura que \texttt{GUILE_LOAD_PATH} contiene la información correcta para que \texttt{guix-daemon} y las herramientas que usa puedan encontrar los módulos Guile que necesitan.}

\begin{verbatim}
$ sudo -E ./pre-inst-env guix-daemon --build-users-group=guixbuild
$ ./pre-inst-env guix build hello
\end{verbatim}

De manera similar, un ejemplo de una sesión de Guile con los módulos Guix disponibles:

\begin{verbatim}
$ ./pre-inst-env guile -c '(use-modules (guix utils)) (pk (%current-system))'
;;; ("x86_64-linux")
\end{verbatim}

... y para un entorno interactivo (REPL) (véase Sección “Using Guile Interactively” en Guile Reference Manual):

\begin{verbatim}
$ ./pre-inst-env guile
scheme@(guile-user)> ,use(guix)
scheme@(guile-user)> ,use(gnu)
scheme@(guile-user)> (define serpientes
  (fold-packages
   (lambda (paquete lst)
     (if (string-prefix? "python"
                  (package-name paquete))
       (cons paquete lst)
       lst))
   '()))
scheme@(guile-user)> (length serpientes)
$1 = 361
\end{verbatim}

El guión pre-inst-env fija todas las variables de entorno necesarias para permitir esto, incluyendo PATH y GUILE_LOAD_PATH.

Fíjese que la orden ./pre-inst-env guix pull no actualiza el árbol de fuentes local; simplemente actualiza el enlace ~/.config/guix/latest (véase Sección 4.6 [Invocación de guix pull], página 51). Ejecute git pull si quiere actualizar su árbol de fuentes local.

14.3 La configuración perfecta

The Perfect Setup to hack on Guix is basically the perfect setup used for Guile hacking (véase Sección “Using Guile in Emacs” en Guile Reference Manual). First, you need more than an editor, you need Emacs (https://www.gnu.org/software/emacs), empowered by the wonderful Geiser (https://nongnu.org/geiser/). To set that up, run:

\begin{verbatim}
guix package -i emacs guile emacs-geiser
\end{verbatim}

Geiser permite desarrollo incremental e interactivo dentro de Emacs: compilación y evaluación de código dentro de los buffers, acceso a documentación en línea (docstrings), completado dependiente del contexto, \texttt{M-}. para saltar a la definición de un objeto, una consola interactiva (REPL) para probar su código, y más (véase Sección “Introducción” en Geiser
Para desarrollar Guix adecuadamente, asegúrese de aumentar la ruta de carga de Guile (load-path) para que encuentre los ficheros fuente de su copia de trabajo:

```
;; Suponiendo que la copia de trabajo de Guix está en ~/src/guix.
(with-eval-after-load 'geiser-guile
  (add-to-list 'geiser-guile-load-path "~/src/guix")
)
```

Para realmente editar el código, Emacs tiene un modo para Scheme muy limpio. Pero además de eso, no debe perderse Paredit (http://www.emacswiki.org/emacs/ParEdit). Provee de facilidades para operar directamente en el árbol sintáctico como elevar una expresión-S o recubrirla, embeber o expulsar la siguiente expresión-S, etc.

We also provide templates for common git commit messages and package definitions in the etc/snippets directory. These templates can be used with YASnippet (https://joaotavora.github.io/yasnippet/) to expand short trigger strings to interactive text snippets. You may want to add the snippets directory to the yas-snippet-dirs variable in Emacs.

```
;; Suponiendo que la copia de trabajo de Guix está en ~/src/guix.
(with-eval-after-load 'yasnippet
  (add-to-list 'yas-snippet-dirs "~/src/guix/etc/snippets")
)
```

Los fragmentos de mensajes de la revisión dependen de Magit (https://magit.vc/) para mostrar los ficheros preparados. En la edición del mensaje de la revisión teclee add seguido de TAB (el tabulador) para insertar la plantilla del mensaje de la revisión de adición de un paquete; teclee update seguido de TAB para insertar una plantilla de actualización de un paquete; teclee https seguido de TAB para insertar una plantilla para cambiar la URI de la página de un paquete a HTTPS.

El fragmento principal para scheme-mode es activado al teclear package... seguido de TAB. Este fragmento también inserta el lanzador origin... que puede ser expandido de nuevo. El fragmento origin puede a su vez insertar otros identificadores de lanzado terminando en ..., que pueden ser expandidos de nuevo.

Additionally we provide insertion and automatic update of a copyright in etc/copyright.el. You may want to set your full name, mail, and load a file.

```
(setq user-full-name "Alice Doe")
(setq user-mail-address "alice@mail.org")
;; Assuming the Guix checkout is in ~/src/guix.
(load-file "~/src/guix/etc/copyright.el")
```

To insert a copyright at the current line invoke M-x guix-copyright.

To update a copyright you need to specify a copyright-names-regexp.

```
(setq copyright-names-regexp
  (format "%s <%s>" user-full-name user-mail-address))
```

You can check if your copyright is up to date by evaluating M-x copyright-update. If you want to do it automatically after each buffer save then add (add-hook 'after-save-hook 'copyright-update) in Emacs.

14.4 Guías de empaquetamiento

La distribución GNU es reciente y puede no disponer de alguno de sus paquetes favoritos. Esta sección describe cómo puede ayudar a hacer crecer la distribución.
Los paquetes de software libre habitualmente se distribuyen en forma de *archivadores de código fuente*—típicamente ficheros tar.gz que contienen todos los ficheros fuente. Añadir un paquete a la distribución significa esencialmente dos cosas: añadir una receta que describe cómo construir el paquete, la que incluye una lista de otros paquetes necesarios para la construcción, y añadir metadatos del paquete junto a dicha receta, como la descripción y la información de licencias.

En Guix toda esta información está contenida en *definiciones de paquete*. Las definiciones de paquete proporcionan una vista de alto nivel del paquete. Son escritas usando la sintaxis del lenguaje de programación Scheme; de hecho, definimos una variable por cada paquete enlazada a su definición y exportamos esa variable desde un módulo (véase Sección 6.1 [Módulos de paquetes], página 76). No obstante, un conocimiento profundo de Scheme no es un pre-requisito para la creación de paquetes. Para más información obre las definiciones de paquetes, véase Sección 6.2 [Definición de paquetes], página 77.

Una vez que una definición de paquete está en su lugar, almacenada en un fichero del árbol de fuentes de Guix, puede probarse usando la orden `guix build` (véase Sección 7.1 [Invocación de guix build], página 119). Por ejemplo, asumiendo que el nuevo paquete se llama gnuexo, puede ejecutar esta orden desde el árbol de construcción de Guix (véase Sección 14.2 [Ejecución de Guix antes de estar instalado], página 436):

```
./pre-inst-env guix build gnuexo --keep-failed
```

El uso de `--keep-failed` facilita la depuración de errores de construcción ya que proporciona acceso al árbol de la construcción fallida. Otra opción útil de línea de órdenes para la depuración es `--log-file`, para acceder al log de construcción.

Si el paquete resulta desconocido para la orden `guix`, puede ser que el fichero fuente contenga un error de sintaxis, o no tenga una cláusula define-public para exportar la variable del paquete. Para encontrar el problema puede cargar el módulo desde Guile para obtener más información sobre el error real:

```
./pre-inst-env guile -c '(use-modules (gnu packages gnuexo))'
```

Una vez que se construya correctamente su paquete, por favor, envíenos un parche (véase Sección 14.6 [Envío de parches], página 447). En cualquier caso, si necesita ayuda también estaremos felices de ayudarle. Una vez el parche se haya incorporado al repositorio de Guix, el nuevo paquete se construye automáticamente en las plataformas disponibles por nuestro sistema de integración continua (ci.guix.gnu.org).

Las usuarias pueden obtener la nueva definición de paquete ejecutando simplemente `guix pull` (véase Sección 4.6 [Invocación de guix pull], página 51). Cuando ci.guix.gnu.org ha terminado de construir el paquete, la instalación del paquete descarga automáticamente los binarios desde allí (véase Sección 4.3 [Sustituciones], página 44). El único lugar donde la intervención humana es necesaria es en la revisión y aplicación del parche.

### 14.4.1 Libertad del software

The GNU operating system has been developed so that users can have freedom in their computing. GNU is *free software*, meaning that users have the four essential freedoms ([https://www.gnu.org/philosophy/free-sw.html](https://www.gnu.org/philosophy/free-sw.html)): to run the program, to study and change the program in source code form, to redistribute exact copies, and to distribute modified versions. Packages found in the GNU distribution provide only software that conveys these four freedoms.
In addition, the GNU distribution follow the free software distribution guidelines (https://www.gnu.org/distros/free-system-distribution-guidelines.html). Among other things, these guidelines reject non-free firmware, recommendations of non-free software, and discuss ways to deal with trademarks and patents.

Algunos paquetes originales, que serían de otra manera software libre, contienen un subconjunto pequeño y opcional que viola estas directrices, por ejemplo debido a que ese subconjunto sea en sí código no-libre. Cuando esto sucede, las partes indeseadas son eliminadas con parches o fragmentos de código en la forma `origin` del paquete (véase Sección 6.2 [Definición de paquetes], página 77). De este modo, `guix build --source` devuelve las fuentes “liberadas” en vez de la versión original de las fuentes.

### 14.4.2 Nombrado de paquetes

Un paquete tiene realmente dos nombres asociados con él: Primero, el nombre de la *variable Scheme* asociada, que aparece después de `define-public`. A través de este nombre, el paquete está disponible en código Scheme, por ejemplo como entrada de otro paquete. Segundo, la cadena en el campo `name` de la definición de paquete. Este nombre se usa por las órdenes de gestión de paquetes como `guix package` y `guix build`.

Ambos normalmente son iguales y corresponden a la conversión a minúsculas del nombre de proyecto elegido por sus creadoras, con los guiones bajos sustituidos por guiones. Por ejemplo, GNUnet está disponible como `gnunet`, y SDL_net como `sdl-net`.

No añadimos prefijos `lib` para paquetes de bibliotecas, a menos que sean parte del nombre oficial del proyecto. Pero vea Sección 14.4.5 [Módulos Python], página 443, y Sección 14.4.6 [Módulos Perl], página 444, para reglas especiales que conciernen a los módulos de los lenguajes Python y Perl.

Los nombres de paquetes de tipografías se manejan de forma diferente, véase Sección 14.4.9 [Tipografías], página 445.

### 14.4.3 Versiones numéricas

Normalmente empaquetamos únicamente la última versión de un proyecto dado de software libre. Pero a veces, por ejemplo para versiones de bibliotecas incompatibles, se necesitan dos (o más) versiones del mismo paquete. Estas necesitan nombres diferentes para las variables Scheme. Usamos el nombre como se define en Sección 14.4.2 [Nombrado de paquetes], página 440, para la versión más reciente; las versiones previas usan el mismo nombre, añadiendo un `-` y el prefijo menor del número de versión que permite distinguir las dos versiones.

El nombre dentro de la definición de paquete es el mismo para todas las versiones de un paquete y no contiene ningún número de versión.

Por ejemplo, las versiones 2.24.20 y 3.9.12 de GTK+ pueden empaquetarse como sigue:

```scheme
(define-public gtk+
  (package
   (name "gtk+")
   (version "3.9.12")
   ...))

(define-public gtk+-2
  (package
```
Capítulo 14: Contribuir

Si también deseásemos GTK+3.8.2, se empaquetaría como

```lisp
(define-public gtk+-3.8
  (package
    (name "gtk+")
    (version "3.8.2")
    ...
  )
)
```

De manera ocasional, empaquetamos instantáneas del sistema de control de versiones (VCS) de las desarrolladoras originales en vez de publicaciones formales. Esto debería permanecer como algo excepcional, ya que son las desarrolladoras originales quienes deben clarificar cuál es la entrega estable. No obstante, a veces es necesario. Por tanto, ¿qué deberíamos poner en el campo `version`?

Claramente, tenemos que hacer visible el identificador de la revisión en el VCS en la cadena de versión, pero también debemos asegurarnos que la cadena de versión incrementa monotónicamente de manera que `guix package --upgrade` pueda determinar qué versión es más moderna. Ya que los identificadores de revisión, notablemente en Git, no incrementan monotónicamente, añadimos un número de revisión que se incrementa cada vez que actualizamos a una nueva instantánea. La versión que resulta debería ser así:

```text
2.0.11-3.cabba9e
```

```
| | '--- ID de revisión original
| |'--- revisión del paquete Guix
| último versión de publicación
```

Es una buena idea recortar los identificadores de revisión en el campo `version` a, digamos, 7 dígitos. Esto evita una molestia estética (asumiendo que la estética tiene importancia aquí) así como problemas relacionados con los límites del sistema operativo como la longitud máxima de una cadena de ejecución #! (127 bytes en el núcleo Linux). Es mejor usar el identificador de revisión completo en `origin`, no obstante, para evitar ambigüedades. Una definición típica de paquete sería así:

```lisp
(define mi-paquete
  (let ((commit "c3f29bc928d5900971f65965feaae59e1272a3f7")
        (revision "1"); Revisión Guix del paquete
    (package
      (version (git-version "0.9" revision commit))
      (source (origin
        (method git-fetch)
        (uri (git-reference
          (url "git://example.org/mi-paquete.git")
          (commit commit)))
        (sha256 (base32 "1mbikn..."))
        (file-name (git-file-name name version)))))
```

14.4.4 Sinopsis y descripciones

Como hemos visto previamente, cada paquete en GNU Guix incluye una sinopsis y una descripción (véase Sección 6.2 [Definición de paquetes], página 77). Las sinopses y descripciones son importantes: son en lo que guix package --search busca, y una pieza crucial de información para ayudar a las usuarias a determinar si un paquete dado cubre sus necesidades. Consecuentemente, las empaquetadoras deben prestar atención a qué se incluye en ellas.

Las sinopses deben empezar con mayúscula y no deben terminar con punto. No deben empezar con un artículo que habitualmente no aporta nada; por ejemplo, se prefiere “Herramienta para chiribizar” sobre “Una herramienta que chiribiza ficheros”. La sinopsis debe decir qué es el paquete—por ejemplo, “Utilidades básicas GNU (ficheros, texto, shell)”—o para qué se usa—por ejemplo, la sinopsis de GNU grep es “Imprime líneas aceptadas por un patrón”.

Tenga en cuenta que las sinopses deben tener un claro significado para una audiencia muy amplia. Por ejemplo, “Manipula la alineación en el formato SAM” puede tener sentido para una investigadora de bioinformática con experiencia, pero puede ser de poca ayuda o incluso llevar a confusión a una audiencia no-especializada. Es una buena idea proporcionar una sinopsis que da una idea del dominio de aplicación del paquete. En ese ejemplo, esto podría ser algo como “Manipula la alineación de secuencias de nucleótidos”, lo que con suerte proporcionará a la usuaria una mejor idea sobre si esto es lo que está buscando.

Las descripciones deben tener entre cinco y diez líneas. Use frases completas, y evite usar acrónimos sin introducirlos previamente. Por favor evite frases comerciales como “líder mundial”, “de potencia industrial” y “siguiente generación”, y evite superlativos como “el más avanzado”—no son útiles para las usuarias que buscan un paquete e incluso pueden sonar sospechosas. En vez de eso, intente ceñirse a los hechos, mencionando casos de uso y características.

Las descripciones pueden incluir marcado Texinfo, lo que es útil para introducir ornamentos como @code o @dfn, listas de puntos o enlaces (véase Sección “Overview” en GNU Texinfo). Por consiguiente, debe ser cuidadosa cuando use algunos caracteres, por ejemplo ‘@’ y llaves, que son los caracteres especiales básicos en Texinfo (véase Sección “Special Characters” en GNU Texinfo). Las interfaces de usuario como guix package --show se encargan de su correcta visualización.

Synopses and descriptions are translated by volunteers at the Translation Project (https://translationproject.org/domain/guix-packages.html) so that as many users as possible can read them in their native language. User interfaces search them and display them in the language specified by the current locale.

Para permitir a xgettext extraerlas como cadenas traducibles, las sinopsis y descripciones deben ser cadenas literales. Esto significa que no puede usar string-append o format para construir estas cadenas:

```latex
(package
  ;; ...
  (synopsis "Esto es traducible")
)
La traducción requiere mucho trabajo, por lo que, como empaquetadora, le rogaríamos que ponga incluso más atención a sus sinopsis y descripciones ya que cada cambio puede suponer trabajo adicional para las traductoras. Para ayudarlas, es posible hacer recomendaciones o instrucciones insertando comentarios especiales como este (véase Sección “xgettext Invocation” en GNU Gettext):

```
;; TRANSLATORS: "X11 resize-and-rotate" should not be translated.
(description "ARandR is designed to provide a simple visual front end for the X11 resize-and-rotate (RandR) extension. ...")
```

14.4.5 Módulos Python

Actualmente empaquetamos Python 2 y Python 3, bajo los nombres de variable Scheme python-2 y python como se explica en Sección 14.4.3 [Versiones numéricas], página 440. Para evitar confusiones y conflictos de nombres con otros lenguajes de programación, parece deseable que el nombre de paquete para un módulo Python contenga la palabra python.

Some modules are compatible with only one version of Python, others with both. If the package Foo is compiled with Python 3, we name it python-foo. If it is compiled with Python 2, we name it python2-foo. Packages should be added when they are necessary; we don’t add Python 2 variants of the package unless we are going to use them.

Si un proyecto ya contiene la palabra python, la eliminamos; por ejemplo, el módulo python-dateutil se empaqueta con los nombres python-dateutil y python2-dateutil. Si el nombre del proyecto empieza con py (por ejemplo pytz), este se mantiene y el prefijo es el especificado anteriormente.

14.4.5.1 Especificación de dependencias

La información de dependencias para paquetes Python está disponible habitualmente en el árbol de fuentes, con varios grados de precisión: en el fichero setup.py, en requirements.txt o en tox.ini.

Su misión, cuando escriba una receta para un paquete Python, es asociar estas dependencias con el tipo apropiado de “entrada” (véase Sección 6.2.1 [Referencia de package], página 81). Aunque el importador de pypi normalmente hace un buen trabajo (véase Sección 7.5 [Invocación de guix import], página 131), puede querer comprobar la siguiente lista para determinar qué dependencia va dónde.

- Actualmente empaquetamos con setuptools y pip instalados como Python 3.4 tiene por defecto. Por tanto no necesita especificar ninguno de ellos como entrada. guix lint le avisará si lo hace.
- Las dependencias Python requeridas en tiempo de ejecución van en propagated-inputs. Típicamente están definidas con la palabra clave install_requires en setup.py, o en el fichero requirements.txt.
- Los paquetes Python requeridos únicamente durante la construcción—por ejemplo, aquellos listados con la palabra clave setup_requires en setup.py—o únicamente para pruebas—por ejemplo, aquellos en tests_require—van en native-inputs. La razón es que (1) no necesitan ser propagados ya que no se requieren en tiempo de ejecución, y (2) en un entorno de compilación cruzada lo que necesitamos es la entrada “nativa"
Ejemplos son las bibliotecas de pruebas pytest, mock y nose. Por supuesto, si alguno de estos paquetes también se necesita en tiempo de ejecución, necesita ir en propagated-inputs.

- Todo lo que no caiga en las categorías anteriores va a inputs, por ejemplo programas o bibliotecas C requeridas para construir los paquetes Python que contienen extensiones C.

- Si un paquete Python tiene dependencias opcionales (extras_require), queda en su mano decidir si las añade o no, en base a la relación utilizad/sobrecarga (véase Sección 14.6 [Envío de parches], página 447).

14.4.6 Módulos Perl

Los programas ejecutables Perl se nombran como cualquier otro paquete, mediante el uso del nombre oficial en minúsculas. Para paquetes Perl que contienen una única clase, usamos el nombre en minúsculas de la clase, substituyendo todas las ocurrencias de :: por guiones y agregando el prefijo perl-. Por tanto la clase XML::Parser se convierte en perl-xml-parser. Los módulos que contienen varias clases mantienen su nombre oficial en minúsculas y también se agrega perl- al inicio. Dichos módulos tienden a tener la palabra perl en alguna parte de su nombre, la cual se elimina en favor del prefijo. Por ejemplo, libwww-perl se convierte en perl-libwww.

14.4.7 Paquetes Java

Los programas Java ejecutables se nombran como cualquier otro paquete, mediante el uso del nombre oficial en minúsculas.

Para evitar confusión y colisiones de nombres con otros lenguajes de programación, es deseable que el nombre del paquete para un paquete Java contenga el prefijo java-. Si el proyecto ya tiene la palabra java, eliminamos esta; por ejemplo, el paquete ngsjaga se empaqueta bajo el nombre java-ngs.

Para los paquetes Java que contienen una clase única o una jerarquía pequeña, usamos el nombre de clase en minúsculas, substituyendo todas las ocurrencias de . por guiones y agregando el prefijo java-. Por tanto la clase apache.commons.cli se convierte en el paquete java-apache-commons-cli.

14.4.8 Rust Crates

Rust programs standing for themselves are named as any other package, using the lowercase upstream name.

To prevent namespace collisions we prefix all other Rust packages with the rust- prefix. The name should be changed to lowercase as appropriate and dashes should remain in place.

In the rust ecosystem it is common for multiple incompatible versions of a package to be used at any given time, so all packages should have a versioned suffix. If a package has passed version 1.0.0 then just the major version number is sufficient (e.g. rust-clap-2), otherwise the version suffix should contain both the major and minor version (e.g. rust-rand-0.6).

Because of the difficulty in reusing rust packages as pre-compiled inputs for other packages the Cargo build system (véase Sección 6.3 [Sistemas de construcción], página 85) presents the #:cargo-inputs and cargo-development-inputs keywords as build system arguments. It would be helpful to think of these as similar to propagated-inputs.
and native-inputs. Rust dependencies and build-dependencies should go in #:cargo-inputs, and dev-dependencies should go in #:cargo-development-inputs. If a Rust package links to other libraries then the standard placement in inputs and the like should be used.

Care should be taken to ensure the correct version of dependencies are used; to this end we try to refrain from skipping the tests or using #:skip-build? when possible. Of course this is not always possible, as the package may be developed for a different Operating System, depend on features from the Nightly Rust compiler, or the test suite may have atrophied since it was released.

14.4.9 Tipografías

Para tipografías que no se instalan generalmente por una usuaria para propósitos tipográficos, o que se distribuyen como parte de un paquete de software más grande, seguimos las reglas generales de empaquetamiento de software; por ejemplo, esto aplica a las tipografías distribuidas como parte del sistema X.Org o las tipografías que son parte de TeX Live.

Para facilitar a las usuarias la búsqueda de tipografías, los nombres para otros paquetes que contienen únicamente tipografías se construyen como sigue, independientemente del nombre de paquete oficial.

El nombre de un paquete que contiene únicamente una familia tipográfica comienza con font-; seguido por el nombre de la tipografía y un guión si la tipografía es conocida, y el nombre de la familia tipográfica, donde los espacios se sustituyen por guiones (y como es habitual, todas las letras mayúsculas se transforman a minúsculas). Por ejemplo, la familia de tipografías Gentium de SIL se empaqueta bajo el nombre de font-sil-gentium.

Para un paquete que contenga varias familias tipográficas, el nombre de la colección se usa en vez del nombre de la familia tipográfica. Por ejemplo, las tipografías Liberation consisten en tres familias: Liberation Sans, Liberation Serif y Liberation Mono. Estas se podrían empaquetar por separado bajo los nombres font-liberation-sans, etcétera; pero como se distribuyen de forma conjunta bajo un nombre común, preferimos empaquetarlas conjuntamente como font-liberation.

En el caso de que varios formatos de la misma familia o colección tipográfica se empaqueten de forma separada, una forma corta del formato, precedida por un guión, se añade al nombre del paquete. Usamos -ttf para tipografías TrueType, -otf para tipografías OpenType y -type1 para tipografías Tipo 1 PostScript.

14.5 Estilo de codificación

En general nuestro código sigue los Estándares de codificación GNU (véase GNU Coding Standards). No obstante, no dicen mucho de Scheme, así que aquí están algunas reglas adicionales.

14.5.1 Paradigma de programación

El código scheme en Guix está escrito en un estilo puramente funcional. Una excepción es el código que incluye entrada/salida, y procedimientos que implementan conceptos de bajo nivel, como el procedimiento memoize.
14.5.2 Módulos
Los módulos Guile que están destinados a ser usados en el lado del constructor deben encontrar en el espacio de nombres (guix build ...). No deben hacer referencia a otros módulos Guix o GNU. No obstante, no hay problema en usar un módulo del lado del constructor en un módulo “del lado del cliente”.

Los módulos que tratan con el sistema GNU más amplio deben estar en el espacio de nombres (gnu ...) en vez de en (guix ...).

14.5.3 Tipos de datos y reconocimiento de patrones
La tendencia en el Lisp clásico es usar listas para representar todo, y recorrerlas “a mano” usando car, cdr, cadr y compañía. Hay varios problemas con este estilo, notablemente el hecho de que es difícil de leer, propenso a errores y una carga para informes adecuados de errores de tipado.

El código de Guix debe definir tipos de datos apropiados (por ejemplo, mediante el uso define-record-type*) en vez de abusar de las listas. Además debe usarse el reconocimiento de patrones, vía el módulo de Guile (ice-9 match), especialmente cuando se analizan listas.

14.5.4 Formato del código
When writing Scheme code, we follow common wisdom among Scheme programmers. In general, we follow the Riastradh’s Lisp Style Rules (https://mumble.net/~campbell/scheme/style.txt). This document happens to describe the conventions mostly used in Guile’s code too. It is very thoughtful and well written, so please do read it.

Algunas formas especiales introducidas en Guix, como el macro substitute* tienen reglas de indentación especiales. Estas están definidas en el fichero .dir-locals.el, el cual Emacs usa automáticamente. Fíjese que además Emacs-Guix proporciona el modo guix-devel-mode que indenta y resalta adecuadamente el código de Guix (véase Sección “Desarrollo” en The Emacs-Guix Reference Manual).

Si no usa Emacs, por favor asegúrese de que su editor conoce esas reglas. Para indentar automáticamente una definición de paquete también puede ejecutar:

./etc/indent-code.el gnu/packages/fichero.scm paquete

Esto indenta automáticamente la definición de paquete en gnu/packages/fichero.scm ejecutando Emacs en modo de procesamiento de lotes. Para indentar un fichero completo, omita el segundo parámetro:

./etc/indent-code.el gnu/services/fichero.scm


Requerimos que todos los procedimientos del nivel superior tengan una cadena de documentación. Este requisito puede relajarse para procedimientos simples privados en el espacio de nombres (guix build ...) no obstante.

Los procedimientos no deben tener más de cuatro parámetros posicionales. Use parámetros con palabras clave para procedimientos que toman más de cuatro parámetros.
14.6 Envío de parches

El desarrollo se lleva a cabo usando el sistema de control de versiones distribuido Git. Por lo tanto, no es estrictamente necesario el acceso al repositorio. Son bienvenidas las contribuciones en forma de parches como los producidos por `git format-patch` enviadas a la lista de correo `guix-patches@gnu.org`. Las desarrolladoras de Guix que lleven un tiempo en ello puede que también quieran leer la sección sobre el acceso al repositorio (véase Sección 14.8 [Acceso al repositorio], página 450).

Esta lista de correo está respaldada por una instancia de Debugs accesible en `https://bugs.gnu.org/guix-patches`, la cual nos permite mantener el seguimiento de los envíos. A cada mensaje enviado a esa lista de correo se le asigna un número de seguimiento; la gente puede realizar aportaciones sobre el tema mediante el envío de correos electrónicos a `NNN@debbugs.gnu.org`, donde NNN es el número de seguimiento (véase [Envío de una serie de parches], página 450).

Le rogamos que escriba los mensajes de revisiones en formato ChangeLog (véase Sección “Change Logs” en GNU Coding Standards); puede comprobar la historia de revisiones en busca de ejemplos.

Antes de enviar un parche que añade o modifica una definición de un paquete, por favor recorra esta lista de comprobaciones:

1. Si las autoras del paquete software proporcionan una firma criptográfica para el archivo de la versión, haga un esfuerzo para verificar la autenticidad del archivo. Para un fichero de firma GPG separado esto puede hacerse con la orden `gpg --verify`.

2. Dedique algún tiempo a proporcionar una sinopsis y descripción adecuadas para el paquete. Véase Sección 14.4.4 [Sinopsis y descripciones], página 442, para algunas directrices.

3. Ejecute `guix lint paquete`, donde `paquete` es el nombre del paquete nuevo o modificado, y corrija cualquier error del que informe (véase Sección 7.7 [Invocación de guix lint], página 141).

4. Asegúrese de que el paquete compile en su plataforma, usando `guix build package`.

5. También le recomendamos que pruebe a construir el paquete en otras plataformas disponibles. Como puede no disponer de acceso a dichas plataformas hardware físicamente, le recomendamos el uso de `qemu-binfmt-service-type` para emularlas. Para activarlo, añada el siguiente servicio a la lista de servicios en su configuración `operating-system`:

   ```scheme
   (service qemu-binfmt-service-type
    (qemu-binfmt-configuration
     (platforms (lookup-qemu-platforms "arm" "aarch64" "mips64el"))
     (guix-support? #t)))
   ```

   Una vez hecho esto, reconfigure su sistema.

Entonces podrá construir paquetes para diferentes plataformas mediante la opción `--system`. Por ejemplo, para la construcción del paquete "hello" para las arquitecturas armhf, aarch64 o mips64 ejecutaría las siguientes órdenes, respectivamente:

```
guix build --system=armhf-linux --rounds=2 hello
guix build --system=aarch64-linux --rounds=2 hello
guix build --system=mips64el-linux --rounds=2 hello
```
6. Asegúrese de que el paquete no usa copias empaquetadas de software ya disponible como paquetes separados.

A veces, paquetes incluyen copias embebidas del código fuente de sus dependencias para conveniencia de las usuarias. No obstante, como distribución, queremos asegurar que dichos paquetes efectivamente usan la copia que ya tenemos en la distribución si hay ya una. Esto mejora el uso de recursos (la dependencia es construida y almacenada una sola vez), y permite a la distribución hacer cambios transversales como aplicar actualizaciones de seguridad para un software dado en un único lugar y que afecte a todo el sistema—algo que esas copias embebidas impiden.

7. Eche un vistazo al perfil mostrado por guix size (véase Sección 7.8 [Invocación de guix size], página 143). Esto le permitirá darse cuenta de referencias a otros paquetes retenidas involuntariamente. También puede ayudar a determinar si se debe dividir el paquete (véase Sección 4.4 [Paquetes con múltiples salidas], página 47), y qué dependencias opcionales deben usarse. En particular, evite añadir texlive como una dependencia: debido a su tamaño extremo, use texlive-tiny o texlive-union.

8. Para cambios importantes, compruebe que los paquetes dependientes (si aplica) no se ven afectados por el cambio; guix refresh --list-dependent package le ayudará a hacerlo (véase Sección 7.6 [Invocación de guix refresh], página 136).

En base al número de paquetes dependientes y, por tanto, del tamaño de la reconstrucción inducida, los revisiones van a ramas separadas, según estas líneas:

- 300 paquetes dependientes o menos
  - rama master (cambios no disruptivos).
- entre 300 y 1.200 paquetes dependientes
  - rama staging (cambios no disruptivos). Esta rama está pensada para ser incorporada en master cada 3 semanas más o menos. Ramas temáticas (por ejemplo, una actualización de la pila de GNOME) pueden ir en una rama específica (digamos, gnome-updates).
- más de 1.200 paquetes dependientes
  - rama core-updates (puede incluir cambios mayores y potencialmente disruptivos). Esta rama está pensada para ser incluida en master cada 2,5 más o menos.

Todas estas ramas son seguidas por nuestra granja de construcción (ci.guix.gnu.org) e incluidas en master una vez todo se ha construido satisfactoriamente. Esto nos permite corregir errores antes de que afecten a usuarias, y reducir la ventana durante la cual los binarios preconstruidos no están disponibles.

Generalmente, ramas distintas a master se consideran congeladas si ha habido una evaluación reciente, o hay una rama -next correspondiente. Por favor, pregunte en la lista de correo o en IRC si no está segura de dónde colocar un parche.

9. Compruebe si el proceso de construcción de un paquete es determinista. Esto significa típicamente comprobar si una construcción independiente del paquete ofrece exactamente el mismo resultado que usted obtuvo, bit a bit.

Una forma simple de hacerlo es construyendo el mismo paquete varias veces seguidas en su máquina (véase Sección 7.1 [Invocación de guix build], página 119):

```
guix build --rounds=2 mi-paquete
```
Capítulo 14: Contribuir

Esto es suficiente una clase común de problemas de no-determinismo, como las marcas de tiempo o salida generada aleatoriamente en el resultado de la construcción.

Otra opción es el uso de guix challenge (véase Sección 7.11 [Invocación de guix challenge], página 153). Puede ejecutarse una vez la revisión del paquete haya sido publicada y construida por ci.guix.gnu.org para comprobar si obtuvo el mismo resultado que usted. Mejor aún: encuentre otra máquina que pueda construirlo y ejecute guix publish. Ya que la máquina remota es probablemente diferente a la suya, puede encontrar problemas de no-determinismo relacionados con el hardware—por ejemplo, el uso de un conjunto de instrucciones extendido diferente—o con el núcleo del sistema operativo—por ejemplo, dependencias en uname o ficheros /proc.

10. Cuando escriba documentación, por favor use construcciones neutrales de género para referirse a la gente\footnote{NdT: En esta traducción se ha optado por usar el femenino para referirse a personas, ya que es el género gramatical de dicha palabra. Aunque las construcciones impersonales pueden adoptarse en la mayoría de casos, también pueden llegar a ser muy artificiales en otros usos del castellano; en ocasiones son directamente imposibles. Algunas construcciones que proponen la neutralidad de género dificultan la lectura automática (-x), o bien dificultan la corrección automática (-e), o bien aumentan significativamente la redundancia y reducen del mismo modo la velocidad en la lectura (-as/os, -as y -os). No obstante, la adopción del género neutro heredado del latín, el que en castellano se ha unido con el masculino, como construcción neutral de género se considera inaceptable, ya que sería equivalente al “it” en inglés, nada más lejos de la intención de las autoras originales del texto.}, como singular “they”, “their”, “them” (https://en.wikipedia.org/wiki/Singular_they) y demás.

11. Compruebe que su parche contiene únicamente un conjunto relacionado de cambios. Agrupando cambios sin relación dificulta y ralentiza la revisión.

Ejemplos de cambios sin relación incluyen la adición de varios paquetes, o una actualización de un paquete junto a correcciones a ese paquete.

12. Por favor, siga nuestras reglas de formato de código, posiblemente ejecutando el guión etc/indent-code.el para que lo haga automáticamente por usted (véase Sección 14.5.4 [Formato del código], página 446).

13. Cuando sea posible, use espejos en la URL de las fuentes (véase Sección 7.3 [Invocación de guix download], página 129). Use URL fiables, no generadas. Por ejemplo, los archivos de GitHub no son necesariamente idénticos de una generación a la siguiente, así que en este caso es normalmente mejor clonar el repositorio. No use el campo name en la URL: no es muy útil y si el nombre cambia, la URL probablemente estará mal.

14. Comprueba si Guix se puede construir correctamente (véase Sección 14.1 [Construcción desde Git], página 435) y trata los avisos, especialmente aquellos acerca del uso de símbolos sin definición.

15. Asegúrese de que sus cambios no rompen Guix y simule guix pull con:

```
guix pull --url=/ruta/a/su/copia --profile=/tmp/guix.master
```

Cuando publique un parche en la lista de correo, use ‘[PATCH] ...’ como el asunto. Puede usar su cliente de correo o la orden git send-email (véase [Envío de una serie de parches], página 450). Preferimos recibir los parches en texto plano, ya sea en línea o como adjuntos MIME. Se le recomienda que preste atención por si su cliente de correo cambia algo como los saltos de línea o la indentación, lo que podría potencialmente romper los parches.
Cuando un error es resuelto, por favor cierre el hilo enviando un correo a NNN-done@debbugs.gnu.org.

### Envío de una serie de parches

Cuando envíe una serie de parches (por ejemplo, usando `git send-email`), por favor mande primero un mensaje a guix-patches@gnu.org, y después mande los parches siguientes a NNN@debbugs.gnu.org para asegurarse de que se mantienen juntos. Véase la documentación de Debbugs (https://debbugs.gnu.org/Advanced.html) para más información. Puede instalar `git send-email` con `guix install git:send-email`.

### 14.7 Seguimiento de errores y parches

El seguimiento de los informes de errores y los envíos de parches se realiza con una instancia de Debbugs en https://bugs.gnu.org. Los informes de errores se abren para el “paquete” guix (en la jerga de Debbugs), enviando un correo a bug-guix@gnu.org, mientras que para los envíos de parches se usa el paquete guix-patches enviando un correo a guix-patches@gnu.org (véase Sección 14.6 [Envío de parches], página 447).

Hay disponible una interfaz web (¡en realidad dos interfaces web!) para la navegación por las incidencias:

- https://bugs.gnu.org/guix muestra informes de errores;


Si usa Emacs, puede encontrar más conveniente la interacción con las incidencias mediante debbugs.el, que puede instalar con:

```
guix install emacs-debbugs
```

Por ejemplo, para enumerar todos las incidencias abiertas en guix-patches pulse:

```
C-u M-x debbugs-gnu RET RET guix-patches RET n y
```


### 14.8 Acceso al repositorio

El acceso de escritura al repositorio es conveniente para personas que contribuyen frecuentemente. Cuando lo crea necesario, considere solicitar acceso al repositorio siguiendo estos pasos:

1. Encuentre tres personas que contribuyan al proyecto que puedan respaldarle. Puede ver la lista de personas que contribuyen en https://savannah.gnu.org/project/memberlist.php?group=guix. Cada una de ellas deberá enviar un correo confirmándolo el respaldo a guix-maintainers@gnu.org (un alias privado para el colectivo de personas que mantienen el proyecto), firmado con su clave OpenPGP.

³ The web interface at https://issues.guix.gnu.org (más agradable) implementada con Mumi, un interesante software escrito en Guile, ¡y en el que puede ayudar! Véase https://git.elephly.net/gitweb.cgi?p=software/mumi.git.
Se espera que dichas personas hayan tenido algunas interacciones con usted en sus contribuciones y sean capaces de juzgar si es suficientemente familiar con las prácticas del proyecto. No es un juicio sobre el valor de su trabajo, por lo que un rechazo debe ser interpretado más bien como un “habrá que probar de nuevo más adelante”.

2. Envíe un correo a guix-maintainers@gnu.org expresando su intención, enumerando a las tres contribuidoras que respaldan su petición, firmado con su clave OpenPGP que usará para firmar las revisiones, y proporcionando su huella dactilar (véase a continuación). Véase https://emailselfdefense.fsf.org/es/ para una introducción a la criptografía de clave pública con GnuPG.

3. Las personas que mantienen el proyecto decidirán en última instancia si conceder o no el acceso de escritura, habitualmente siguiendo las recomendaciones de las personas de referencia proporcionadas.

4. Una vez haya conseguido acceso, en caso de hacerlo, por favor envíe un mensaje a guix-devel@gnu.org para notificarlo, de nuevo firmado con la clave OpenPGP que vaya a usar para firmar las revisiones (hágalo antes de subir su primera revisión). De esta manera todo el mundo puede enterarse y asegurarse de que controla su clave OpenPGP.

5. Asegúrese de leer el resto de esta sección y... ¡a disfrutar!

Nota: Quienes mantienen el proyecto están encantadas de proporcionar acceso al repositorio a personas que han contribuido durante algún tiempo y tienen buen registro—¡no sea tímida y no subestime su trabajo!

No obstante, tenga en cuenta que el proyecto está trabajando hacia la automatización de la revisión de parches y el sistema de mezclas, lo que, como consecuencia, puede hacer necesario que menos gente tenga acceso de escritura al repositorio principal. ¡Seguiremos informando!

Si obtiene acceso, por favor asegúrese de seguir la política descrita a continuación (el debate sobre dicha política puede llevarse a cabo en guix-devel@gnu.org).

Los parches no triviales deben enviarse siempre a guix-patches@gnu.org (los parches triviales incluyen la corrección de errores tipográficos, etcétera). Esta lista de correo rellena la base de datos de seguimiento de parches (véase Sección 14.7 [Seguimiento de errores y parches], página 450).

Para los parches que únicamente añaden un nuevo paquete, y uno simple, está bien subirlos directamente, si tiene confianza en ello (lo que significa que lo ha construido de manera correcta en un entorno chroot, y ha hecho un auditado razonable de derechos de copia y licencias). Lo mismo puede ser dicho de las actualizaciones de paquetes, excepto actualizaciones que desencadenen muchas reconstrucciones (por ejemplo, la actualización de GnuTLS o GLib). Tenemos una lista de correo para las notificaciones de revisiones (guix-commits@gnu.org), de manera que la gente pueda enterarse. Antes de subir los cambios, asegúrese de ejecutar `git pull --rebase`.

Todas las revisiones que se suban al repositorio central de Savannah deben estar firmadas por una clave OpenPGP, y la clave pública debe subirse a su cuenta de usuaria en Savannah y a servidores públicos de claves, como keys.openpgp.org. Para configurar que Git firme automáticamente las revisiones ejecute:

    `git config commit.gpgsign true`
git config user.signingkey CABBA6EA1DC0FF33

Puede evitar la subida accidental de revisiones sin firma a Savannah mediante el uso del hook pre-push de Git que se encuentra en etc/git/pre-push:

    cp etc/git/pre-push .git/hooks/pre-push

Cuando suba un commit en nombre de alguien, por favor añada una línea de Signed-off-by al final del mensaje de la revisión—por ejemplo con `git am --signoff`. Esto mejora el seguimiento sobre quién hizo qué.

Para cualquier otra cosa, envíe un mensaje a guix-patches@gnu.org y deje tiempo para que sea revisado, sin subir ningún cambio (véase Sección 14.6 [Envío de parches], página 447). Si no recibe ninguna respuesta después de dos semanas y tiene confianza en ello, está bien subir el cambio.

Esta última parte está sujeta a revisión, para permitir a individualidades que suban cambios que no puedan generar controversia directamente en partes con las que estén familiarizadas.

Una última cosa: el proyecto sigue adelante porque las contribuidoras no solo suben sus cambios, sino que también ofrecen su tiempo revisando y subiendo cambios de otras personas. Como contribuidora, también se agradece que use su experiencia y derechos de escritura en el repositorio para ayudar a otras personas que quieren contribuir.
15 Reconocimientos

Guix está basado en el gestor de paquetes Nix (https://nixops.org/nix), que fue diseñado e implementado por Eelco Dolstra, con contribuciones de otra gente (véase el fichero nix/AUTHORS en Guix). Nix fue pionero en la gestión de paquetes funcional, y promovió características sin precedentes, como las actualizaciones de paquetes transaccionales y vuelta atrás, perfiles por usuaria y un proceso de compilación referencialmente transparente. Sin este trabajo, Guix no existiría.

Las distribuciones de software basadas en Nix, Nixpkgs y NixOS, también han sido una inspiración para Guix.

GNU Guix en sí es un trabajo colectivo con contribuciones de un número de gente. Mire el fichero AUTHORS en Guix para más información sobre esa gente maja. El fichero THANKS enumera personas que han ayudado informando de errores, se han encargado de infraestructura, han proporcionando arte y temas, han realizado sugerencias, y más—¡gracias!
Apéndice A Licencia de documentación libre GNU

Version 1.3, 3 November 2008
https://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released
under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING
You may copy and distribute the Document in any medium, either commercially or
noncommercially, provided that this License, the copyright notices, and the license
notice saying this License applies to the Document are reproduced in all copies, and
that you add no other conditions whatsoever to those of this License. You may not use
technical measures to obstruct or control the reading or further copying of the copies
you make or distribute. However, you may accept compensation in exchange for copies.
If you distribute a large enough number of copies you must also follow the conditions
in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly
display copies.

3. COPYING IN QUANTITY
If you publish printed copies (or copies in media that commonly have printed covers) of
the Document, numbering more than 100, and the Document’s license notice requires
Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all
these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on
the back cover. Both covers must also clearly and legibly identify you as the publisher
of these copies. The front cover must present the full title with all words of the title
equally prominent and visible. You may add other material on the covers in addition.
Copying with changes limited to the covers, as long as they preserve the title of the
Document and satisfy these conditions, can be treated as verbatim copying in other
respects.
If the required texts for either cover are too voluminous to fit legibly, you should put
the first ones listed (as many as fit reasonably) on the actual cover, and continue the
rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100,
you must either include a machine-readable Transparent copy along with each Opaque
copy, or state in or with each Opaque copy a computer-network location from which
the general network-using public has access to download using public-standard network
protocols a complete Transparent copy of the Document, free of added material. If
you use the latter option, you must take reasonably prudent steps, when you begin
distribution of Opaque copies in quantity, to ensure that this Transparent copy will
remain thus accessible at the stated location until at least one year after the last time
you distribute an Opaque copy (directly or through your agents or retailers) of that
edition to the public.
It is requested, but not required, that you contact the authors of the Document well
before redistributing any large number of copies, to give them a chance to provide you
with an updated version of the Document.

4. MODIFICATIONS
You may copy and distribute a Modified Version of the Document under the conditions
of sections 2 and 3 above, provided that you release the Modified Version under precisely
this License, with the Modified Version filling the role of the Document, thus licensing
distribution and modification of the Modified Version to whoever possesses a copy of
it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the
   Document, and from those of previous versions (which should, if there were any,
be listed in the History section of the Document). You may use the same title as a
previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for
authorship of the modifications in the Modified Version, together with at least five
of the principal authors of the Document (all of its principal authors, if it has fewer
than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the
publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other
copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public
permission to use the Modified Version under the terms of this License, in the form
shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover
Texts given in the Document’s license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item
stating at least the title, year, new authors, and publisher of the Modified Version as
given on the Title Page. If there is no section Entitled “History” in the Document,
create one stating the title, year, authors, and publisher of the Document as given
on its Title Page, then add an item describing the Modified Version as stated in
the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to
a Transparent copy of the Document, and likewise the network locations given in
the Document for previous versions it was based on. These may be placed in the
“History” section. You may omit a network location for a work that was published
at least four years before the Document itself, or if the original publisher of the
version it refers to gives permission.
K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title
of the section, and preserve in the section all the substance and tone of each of the
contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and
in their titles. Section numbers or the equivalent are not considered part of the
section titles.
M. Delete any section Entitled “Endorsements”. Such a section may not be included
in the Modified Version.
N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in
title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify
as Secondary Sections and contain no material copied from the Document, you may at
your option designate some or all of these sections as invariant. To do this, add their
titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.
7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.
10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.
ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the
document and put the following copyright and license notices just after the title page:

Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ‘‘GNU
Free Documentation License’’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the
“with...Texts.” line with this:

with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.

If you have Invariant Sections without Cover Texts, or some other combination of the
three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing
these examples in parallel under your choice of free software license, such as the GNU
General Public License, to permit their use in free software.
Índice de conceptos

A

AArch64, cargadores de arranque .................................... 396
acceso al escáner .................................................. 244
acceso al repositorio, para desarrolladoras ..................... 450
acceso remoto al daemon .......................................... 19, 99
acceso SSH a los daemons de construcción .................... 99
acciones, de servicios de Shepherd .............................. 421
acento grave (quasiquote) ....................................... 79
ACL (listas de control de acceso), para sustituciones ............ 45
actualización del daemon de Guix, en una distribución distinta .................................................. 22
actualizaciones de seguridad ...................................... 425
actualizar Guix ....................................................... 51
actualizar Guix para la usuaria root, en una distribución distinta .................................................. 22
Actualizar Guix, en una distribución distinta .................... 22
actualizar la versión de Guix ...................................... 51
actualizar paquetes .................................................. 39
acuerdo de contribución ............................................ 435
agente para el envío de correo (MTA) .......................... 271
ahorro de espacio .................................................. 402
aislamiento .......................................................... 2
alias de guix package .............................................. 36
alias, para direcciones de correo electrónico .................. 276
almacén .............................................................. 2, 98
ALSA .............................................................. 245
aplicación empaquetada ........................................... 71
archivado de código fuente, Software Heritage ................. 141
archive .................................................................. 63
archivo normalizado (nar) ......................................... 64
ARM, cargadores de arranque .................................. 396
arranque dual ......................................................... 398
arranque EFI ........................................................... 398
arranque obsoleto, en máquinas Intel .......................... 162
arranque por BIOS, en máquinas Intel ......................... 162
arranque UEFI ....................................................... 162
asociación de teclas ................................................ 179
Audit ................................................................. 388
autorizar, archivos .................................................. 65

B

búsqueda de documentación ......................................... 423
bag (representación de paquetes de bajo nivel) .............. 85
bajada de nivel, de objetos de alto nivel en expresiones-G .......................................................... 109, 117
base de datos ........................................................ 247
batería de pruebas .................................................. 8
binarios del lanzamiento inicial .................................. 427, 432
binarios pre-construidos .......................................... 44
binarios reposicionables ........................................... 72
binarios reposicionables, con guix pack ....................... 71
binfmt_misc ......................................................... 363
Bioconductor ......................................................... 132
BIOS, cargador de arranque ..................................... 396
bootloader ........................................................... 396
borrado de generaciones del sistema ......................... 402
borrado de paquetes .............................................. 36
borrar paquetes ..................................................... 36
buscar paquetes .................................................... 41

código de conducta, de contribuidoras ......................... 435
caché de tipografías ............................................... 21
Cambio de modo (modeswitch) .................................. 202
canales, para paquetes personales .............................. 55
cargador de arranque .............................................. 396
Cargo (sistema de construcción de Rust) ....................... 88
cat-avatar-generator ............................................. 318
CD, formato de imagen ............................................ 405
certificados TLS ................................................... 320
certificados X.509 .................................................. 390
channels ............................................................. 54
channels.scm, fichero de configuración ....................... 53, 54
chroot ................................................................. 10, 15
cifrado de disco ................................................... 176
cifrado del intercambio ........................................... 176
clausura ............................................................. 50, 143, 144
clausura de módulos ............................................... 110
clavar, canales ..................................................... 58, 59
claves autorizadas, SSH ......................................... 214
Clojure (lenguaje de programación) .......................... 89
codificación normalizada en los nombres de localizaciones .......................................................... 182
colisiones del perfil ................................................ 41
colisiones, en un perfil ........................................... 41
Índice de conceptos

coma (unquote) ........................................ 79
compartir elementos del almacén
entre máquinas ....................................... 155
compilación cruzada .................................. 74, 80, 109, 126
compilación cruzada,
    dependencias de paquetes ....................... 82
composición de revisiones de Guix ................ 60
comprobación de integridad ........................ 50
confianza, de binarios pre-construidos .......... 47
configuración de guix pull ......................... 54
configuración del sistema .......................... 161
Conman ................................................. 203
construcción de paquetes ........................... 119
construcciones reproducibles ....................... 15, 35, 46, 153
construcciones reproducibles, comprobar ........ 448
construcciones verificables ........................ 153
container ................................................. 69, 156
contenedor, entorno de construcción ............... 15
contraseña, para cuentas de usuario ............... 177
control de acceso mandatorio, SELinux .......... 14
conversión de un paquete ........................... 131
copiar, elementos del almacén, por SSH .......... 155
correo ................................................... 249
correo electrónico (email) .......................... 249
correo electrónico, alias .............................. 276
corrupción, recuperarse de ........................ 51, 127
CPAN ..................................................... 132
CRAN ..................................................... 132
crate ..................................................... 136
Crate ..................................................... 136
Creación de imágenes del sistema en
    varios formatos .................................. 403
creación de literales ................................ 78
cron ..................................................... 196
CTAN ..................................................... 133
cuentas ................................................. 176
cuentas de usuario ................................... 176
CVE, vulnerabilidades y
    exposiciones comunes ............................ 142
D
daemon ................................................... 9
daemon de caché del servicio de nombres ........ 189
daemon de caché del servicio de
    nombres (ncsl) ..................................... 20
daemon de construcción .............................. 2
daemon de IMAP3 de GNU Mailutils ............... 276
daemon Early OOM (early out of memory) ......... 383
daemon, acceso remoto .............................. 19, 99
daemon, configuración en cluster .................. 19, 99
daemons ............................................... 412
dar formato al código ................................ 446
darkstat ............................................... 290
ddcclient .............................................. 333
Debbugs, interfaz web Mumi ......................... 315
Debbugs, sistema de seguimiento
    de incidencias .................................... 450
declaración del perfil ................................ 39
deduplicación ......................................... 18, 51
definición de localización ........................... 181
definición de paquete, edición ...................... 129
delegando trabajo ..................................... 11, 16
dependencias de un paquete ......................... 50, 145
dependencias, canales ................................ 56
dependencias, tiempo de construcción ............. 101
dependencias, tiempo de ejecución ................ 101
derivación .............................................. 50, 76
derivaciones ............................................ 101
derivaciones de salida fija ......................... 101
desarrollo de software ................................ 66
descargando las fuentes de paquetes ............... 129
descargar el binario de Guix ....................... 5
descripción de paquete .............................. 442
deshacer transacciones ................................ 40, 53
determinismo, comprobación ........................ 126
determinismo, del proceso de construcción ........ 448
DHCP ..................................................... 30
DHCP, servicio de red ................................ 200
diccionario ............................................. 386
directorio de estado .................................. 8
directorios relacionados con una
distribución distinta ................................ 5
disco cifrado .......................................... 31, 164
disco inicial de RAM .................................. 169, 393, 394
disponibilidad de sustituciones ..................... 157
dispositivos de intercambio ......................... 170
dispositivos traducidos .............................. 175
distribución Android .................................. 87
Distribución de Sistema Guix,
    ahora sistema Guix ................................ 2
distribución de teclado ............................. 29, 179
distribución de teclado, configuración ............ 180
distribución de teclado, definición ................. 179
distribución de teclado, para el
    gestor de arranque ................................ 397
distribución de teclado, para Xorg ................. 223
distribución distinta ................................ 5, 19
DNS (sistema de nombres de dominio) .............. 323
Docker ................................................... 387
Docker, construir una imagen con guix pack ...... 71
documentación ......................................... 47
documentación, búsqueda ............................ 423
dominio (realm), kerberos ........................... 296
DVD, formato de imagen .............................. 405
E
earlyoom ............................................... 383
EFI, cargador de arranque .......................... 396
EFI, instalación ....................................... 30
el fichero hosts ...................................... 169
elementos del almacén ................................ 98
elementos del almacén no válidos .................. 100
elpa ...................................................... 135
emacs ................................................... 21
empaquetado .......................................... 71
Índice de conceptos

empaquetado de software........................................ 71
empaquetamientos............................................. 448
emulación.................................................... 363
entorno de construcción....................................... 9, 15
entorno interactivo............................................ 437
entorno persistente............................................. 67
entorno, entorno de construcción de paquetes........... 66
entornos de construcción reproducibles............... 66
entornos de desarrollo......................................... 66
entrada de pantalla táctil, para Xorg ................. 385
entrada de tablet, para Xorg................................ 385
entradas propagadas........................................... 37
entradas, de paquetes......................................... 81
entradas, para paquetes Python............................ 443
env, en /usr/bin.............................................. 184
envíos de parches, seguimiento............................ 450
escalado de frecuencia de la CPU con thermal........ 351
escape de código de construcción.......................... 108
ESP, partición del sistema EFI.............................. 30
espacio en disco................................................ 48
estadísticas, para sustituciones............................ 157
estado de delegación.......................................... 14
estilo de codificación........................................ 446
estrategia “on-error”........................................... 406
estrategia de planificación de reconstrucciones.... 448
estrategia de ramas............................................. 448
estrategia en caso de error.................................. 406
estados de código............................................. 108
exportar elementos del almacén............................ 63
expresión-G................................................... 108
extender la colección de paquetes (canales)........ 55
extensibilidad de la distribución........................ 2
extensiones de servicios....................................... 412
extensiones, para expresiones G......................... 111

G

GAD (DAG en Inglés)........................................... 145
GC, añadir raíces.............................................. 127
GC, raíces del recolector de basura.................... 18, 48
GCC............................................................ 22
GDM............................................................ 218
gem............................................................ 132
generaciones.................................................... 40, 43, 53, 401
gestión de energía con TLP................................. 344
gestión de paquetes funcional.............................. 2
gestión de red................................................ 201
gestor de ingreso en el sistema............................ 218, 222
gestor de llenado de la memoria (out of memory killer).383
gestor de ventanas........................................... 218
git send-email............................................... 450
Git, alojamiento.............................................. 379
Git, interfaz web............................................. 366
Git, usar la última revisión................................. 123
git-send-email............................................... 450
GNOME, gestor de ingreso al sistema.................. 218
gpm............................................................. 194
granja de construcción....................................... 44
grupos.......................................................... 177, 178
GSS............................................................ 341
GSSD........................................................... 341
guía del instalador............................................ 5
guix archive................................................... 63
guix build...................................................... 119
guix challenge................................................ 153
guix container............................................... 156
guix download............................................... 129
guix edit....................................................... 129
guix environment............................................ 66
guix graph.................................................... 145
guix hash...................................................... 130
guix lint....................................................... 141
guix publish................................................ 149
guix pull....................................................... 51
guix pull para la usuario root, en una distribución distinta.................. 22
guix pull, fichero de configuración.................... 54
guix refresh................................................... 136
guix size...................................................... 143
guix time-machine......................................... 59
GuixSD, ahora sistema Guix................................ 2

H

hackage......................................................... 134
HDPI.......................................................... 399
HiDPI........................................................ 399
hpguiweb..................................................... 319
HTTP.......................................................... 303
HTTP, HTTPS............................................... 320
HTTPS, certificados........................................ 390
huella dactilar................................................. 384
Índice de conceptos

I
identificación, de una copia de Guix .... 435
idmapd .................................. 342
Imágenes de sistema, creación en
  varios formatos .......................... 403
imagen de instalación ...................... 34
IMAP .................................... 272
importación de un paquete ................. 131
importar paquetes ........................ 131
incompatibilidad, de datos de localización .. 182
indentación, de código .................... 446
inidata .................................. 207
infiniord ................................. 60, 117
Info, formato de documentación .......... 423
informes de errores, seguimiento ......... 450
initrd .................................. 169, 393, 394
injertos (grafts en inglés) ................. 425
inputattach ............................. 385
insert or update copyright ................ 438
instalación de paquetes .................... 36
instalación del sistema Guix ............... 23
instalación por SSH ........................ 30
instalar Guix ................................ 5
instalar Guix desde binarios ............... 5
instalar paquetes .......................... 36
integración continua ....................... 123, 342
integración continua, estadísticas ....... 157
integridad, del almacen ................... 50
interfaces de usuario ...................... 2
invalidación de caché, nscd .............. 189
Invocación de guix import ............... 131
iptables .................................. 204
IRC (Internet Relay Chat) ................. 283, 284
ISO-9660, formato ....................... 405

J
jabber .................................... 276
jackd .................................... 196
java ...................................... 444
JSON ..................................... 63
JSON, importación ........................ 133

K
Kerberos .................................. 295

L
límites por sesión .......................... 196
lanzamiento inicial ......................... 427
ld-wraper ................................ 22
LDAP ..................................... 297
lenguaje de programación Rust ............... 88
Let’s Encrypt ............................. 320
licencia, de paquetes ....................... 83
licencia, GNU Free Documentation License .. 454
lirc ....................................... 385
lista negra, de módulos del núcleo .......... 394
listas de control de acceso (ACL), .. 45
para sustituciones ........................ 45
localización .............................. 181
localizaciones, cuando no se está en ..
  el sistema Guix .......................... 19
logging .................................. 191, 198
logs de construcción, acceso .............. 127
logs de construcción, publicación ....... 150
LUKS ..................................... 176

M
máquina virtual ............................ 402, 410
máquina virtual, instalación del sistema Guix .. 33
módulo, lista negra ......................... 394
módulos de identificación conectables .... 171
módulos importados, para expresiones-G .... 110
mónada ................................... 104
mónada de estado .......................... 106
M-x copyright-update ...................... 438
M-x guix-copyright ......................... 438
manifiesto del perfil ....................... 39
mapa de teclas, para Xorg .................. 223
marcado Texinfo, en
  descripciones de paquetes ............... 442
mcron .................................... 196
menú de arranque ........................ 398
mensaje del día ............................ 185
mensajería ................................ 276
metadatos, canales ......................... 56
ModemManager ............................ 201
mpd ....................................... 352
MTA (agente para el envío de correo) ...... 271
Mumi, interfaz web de Debbuugs ........... 315
Murmur (servidor VoIP) .................... 285

N
número de versión, para revisiones de VCS .. 441
nar, formato de archivo ................... 64
nscd (daemon de caché del
  servicio de nombres) ..................... 20
NetworkManager .......................... 202
NFS ....................................... 339
NFS, servidor ............................. 339
nftables .................................. 205
NIS (servicio de información de red) ....... 20
<table>
<thead>
<tr>
<th>Concepto</th>
<th>Página</th>
</tr>
</thead>
<tbody>
<tr>
<td>nivel de detalle de los mensajes, de las herramientas de línea de órdenes</td>
<td>121</td>
</tr>
<tr>
<td>Nix</td>
<td>388</td>
</tr>
<tr>
<td>no-determinismo, en la</td>
<td></td>
</tr>
<tr>
<td>construcción de paquetes</td>
<td>154</td>
</tr>
<tr>
<td>nombre de localización</td>
<td>182</td>
</tr>
<tr>
<td>nombre de paquete</td>
<td>440</td>
</tr>
<tr>
<td>noticias de los canales</td>
<td>53</td>
</tr>
<tr>
<td>noticias, para canales</td>
<td>57</td>
</tr>
<tr>
<td>nscl, invalidación de caché</td>
<td>189</td>
</tr>
<tr>
<td>nscl, servicio LDAP</td>
<td>297</td>
</tr>
<tr>
<td><strong>nss-certs</strong></td>
<td>21, 390</td>
</tr>
<tr>
<td>nss-mdns</td>
<td>391</td>
</tr>
<tr>
<td>NSS</td>
<td>391</td>
</tr>
<tr>
<td>NSS (selector de servicios de nombres), glibc</td>
<td>20</td>
</tr>
<tr>
<td>nsswitch.conf</td>
<td>20</td>
</tr>
<tr>
<td>NTP (protocolo de tiempo de red), servicio</td>
<td>205</td>
</tr>
<tr>
<td>ntdp, servicio para el daemon del protocolo de tiempo de red NTP</td>
<td>205</td>
</tr>
<tr>
<td><strong>O</strong></td>
<td></td>
</tr>
<tr>
<td>objetos tipo-fichero</td>
<td>113</td>
</tr>
<tr>
<td>OCaml</td>
<td>136</td>
</tr>
<tr>
<td>on-error</td>
<td>406</td>
</tr>
<tr>
<td>oom</td>
<td>383</td>
</tr>
<tr>
<td>OPAM</td>
<td>136</td>
</tr>
<tr>
<td>OpenNTPD</td>
<td>206</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td></td>
</tr>
<tr>
<td>página web oficial</td>
<td>7</td>
</tr>
<tr>
<td>páginas de manual</td>
<td>423</td>
</tr>
<tr>
<td>páginas man</td>
<td>423</td>
</tr>
<tr>
<td>pack</td>
<td>71</td>
</tr>
<tr>
<td>pam-krb5</td>
<td>296</td>
</tr>
<tr>
<td>pam-mount</td>
<td>380</td>
</tr>
<tr>
<td>PAM</td>
<td>171</td>
</tr>
<tr>
<td>paquete, comprobación de errores</td>
<td>141</td>
</tr>
<tr>
<td>paquetes</td>
<td>35</td>
</tr>
<tr>
<td>paquetes con colisiones en perfiles</td>
<td>41</td>
</tr>
<tr>
<td>paquetes de extensión de postgresql</td>
<td>247</td>
</tr>
<tr>
<td>(extension-packages)</td>
<td></td>
</tr>
<tr>
<td>paquetes de salida múltiple</td>
<td>47</td>
</tr>
<tr>
<td>paquetes inferiores</td>
<td>60, 61</td>
</tr>
<tr>
<td>paquetes personales (canales)</td>
<td>55</td>
</tr>
<tr>
<td>paquetes, creación</td>
<td>438</td>
</tr>
<tr>
<td>parches</td>
<td>78</td>
</tr>
<tr>
<td>pasarela IRC</td>
<td>283</td>
</tr>
<tr>
<td>Patchwork</td>
<td>312</td>
</tr>
<tr>
<td>pcscl</td>
<td>384</td>
</tr>
<tr>
<td>perl</td>
<td>37</td>
</tr>
<tr>
<td>perl</td>
<td>444</td>
</tr>
<tr>
<td>personalización, de paquetes</td>
<td>2, 76</td>
</tr>
<tr>
<td>personalización, de servicios</td>
<td>163</td>
</tr>
<tr>
<td>php-fpm</td>
<td>316</td>
</tr>
<tr>
<td>PID 1</td>
<td>420</td>
</tr>
<tr>
<td>pipefs</td>
<td>341</td>
</tr>
<tr>
<td>planificación de trabajos</td>
<td>196</td>
</tr>
<tr>
<td>plantillas</td>
<td>438</td>
</tr>
<tr>
<td>POP</td>
<td>272</td>
</tr>
<tr>
<td>postgis</td>
<td>247</td>
</tr>
<tr>
<td>prioridad</td>
<td>196</td>
</tr>
<tr>
<td>procedimiento de extensión de construcción</td>
<td>11</td>
</tr>
<tr>
<td>programas con setuid</td>
<td>389</td>
</tr>
<tr>
<td>prometheus-node-exporter</td>
<td>291</td>
</tr>
<tr>
<td>propósito</td>
<td>2</td>
</tr>
<tr>
<td>prueba de delegación</td>
<td>13</td>
</tr>
<tr>
<td>pull</td>
<td>51</td>
</tr>
<tr>
<td>PulseAudio, sonido</td>
<td>245</td>
</tr>
<tr>
<td>punto de entrada, para imágenes de Docker</td>
<td>73</td>
</tr>
<tr>
<td>pypi</td>
<td>131</td>
</tr>
<tr>
<td>python</td>
<td>443</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>QEMU</td>
<td>410</td>
</tr>
<tr>
<td>quote</td>
<td>78</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>raíces del recolector de basura</td>
<td>18, 48</td>
</tr>
<tr>
<td>raíces del recolector de basura, añadir</td>
<td>127</td>
</tr>
<tr>
<td>raíces del recolector de basura, para empaquetados</td>
<td>75</td>
</tr>
<tr>
<td>raíz del recolector de basura, para entornos</td>
<td>67</td>
</tr>
<tr>
<td>ratón</td>
<td>194</td>
</tr>
<tr>
<td>recolector de basura, para entornos</td>
<td>205</td>
</tr>
<tr>
<td>reparable</td>
<td>51</td>
</tr>
<tr>
<td>reparar el almacén</td>
<td>127</td>
</tr>
<tr>
<td>reemplazos de paquetes, para injertos</td>
<td>425</td>
</tr>
<tr>
<td>reescritura de la entrada</td>
<td>80</td>
</tr>
<tr>
<td>reescritura del árbol de dependencias</td>
<td>80</td>
</tr>
<tr>
<td>references</td>
<td>101</td>
</tr>
<tr>
<td>registro de construcción, nivel de descripción</td>
<td>121</td>
</tr>
<tr>
<td>reloj de tiempo real</td>
<td>205</td>
</tr>
<tr>
<td>replicación de, entornos de software</td>
<td>36</td>
</tr>
<tr>
<td>replicar Guix</td>
<td>58, 59, 62</td>
</tr>
<tr>
<td>REPL</td>
<td>437</td>
</tr>
<tr>
<td>reproducibilidad</td>
<td>35, 62</td>
</tr>
<tr>
<td>reproducibilidad, comprobación</td>
<td>126</td>
</tr>
<tr>
<td>reproducción, de Guix</td>
<td>58, 59</td>
</tr>
<tr>
<td>resolución</td>
<td>399</td>
</tr>
<tr>
<td>reto (challenge)</td>
<td>153</td>
</tr>
<tr>
<td>rotación de logs</td>
<td>198</td>
</tr>
<tr>
<td>rottlog</td>
<td>198</td>
</tr>
<tr>
<td>rpc_pipes</td>
<td>341</td>
</tr>
<tr>
<td>rpcbind</td>
<td>340</td>
</tr>
<tr>
<td>rust</td>
<td>444</td>
</tr>
<tr>
<td>ruta de búsqueda de módulos de paquetes</td>
<td>76</td>
</tr>
<tr>
<td>ruta de derivación</td>
<td>101</td>
</tr>
<tr>
<td>rutas de búsqueda</td>
<td>37, 40</td>
</tr>
</tbody>
</table>
Índice de conceptos

rutas del almacén .................................................. 98
RYF, Respete Su Libertad .............................. 23

S
salidas ......................................................... 47
salidas del paquete ........................................ 47
secretos, servicio Knot ................................. 331
seguimiento de incidencias ............................... 450
seguimiento de procedencia, de
  artefactos de software ................................. 36
seguimiento de procedencia, del
  sistema operativo ................................ 401, 405, 419
seguridad .................................................... 45
seguridad, guix-daemon ................................. 14
selector de servicios de nombres ................. 391
selector de servicios de nombres, glibc ........ 20
SELinux, instalación de la política .............. 14
SELinux, limitaciones ................................ 14
SELinux, política del daemon ..................... 14
series de parches ......................................... 450
services .................................................. 163, 412
servicio Cgit ............................................... 366
servicio de asociación de nombres ............ 342
Servicio de información de red (NIS) ........ 20
servicio del sistema ................................ 413
servicio Gitolite ......................................... 379
servicio oculto .......................................... 210
servicios de shepherd ................................ 420
servicios del sistema ................................ 183
servicios esenciales .................................. 171
servicios one-shot, para Shepherd .......... 421
servidor de sustituciones ......................... 439
servidor SSH .............................................. 211, 212, 411
servidor virtual privado (VPS) .................. 33
servidor VoIP ........................................... 285
sh, en /bin ............................................... 184
sin cables ................................................. 29, 201
Singularity, construir una imagen
  con guix pack ...................................... 72
sinopsis de paquete .................................. 442
sistema de construcción ............................ 85
Sistema de construcción GNU .................... 78
Sistema de construcción NDK de Android .... 87
sistema de construcción simple de Clojure .... 89
sistema de inicio ...................................... 420
sistema de nombres de dominio (DNS) ...... 323
sistema de seguridad global (GSS) .......... 341
Sistema Guix ............................................. 2, 3
sistema Guix, instalación ......................... 23
sistema X Window ..................................... 218
size ...................................................... 143
SMTP ..................................................... 271
Software Heritage, archivo de código fuente .. 141
software libre ........................................ 439
sonido .................................................. 245
soporte de hardware en el sistema Guix .... 23
soporte de impresión con CUPS ............... 224
spice .................................................... 385
SQL ....................................................... 247
SquashFS, construir una imagen
  con guix pack ..................................... 72
SSH ..................................................... 211, 212, 411
SSH, claves autorizadas ............................ 214
SSH, copiar elementos del almacén .......... 155
stackage ............................................... 135
subdirectorio, canales ............................. 57
sudo y guix pull ..................................... 33
sustituciones ........................................ 16, 36, 44
sustituciones, autorización de las mismas ... 7, 45,
  191
sustituciones, cómo desactivar .................. 45
sysctl ............................................... 384
syslog ................................................. 191

T
tamaño del paquete ................................. 143
Tex Live .............................................. 133
thermad ............................................... 351
tiempo de construcción, dependencias .... 101
tiempo de ejecución, dependencias .......... 101
tiempo real .......................................... 196
tipo de servicio ..................................... 417
tipografías ........................................... 21, 445
tipos de servicio ................................ 413
tipos de sesión (X11) .............................. 218
tlp ..................................................... 344
TLS ................................................... 390
Tor .................................................... 209
traducción de dispositivos ...................... 175
transacciones ........................................ 35, 36
transacciones, deshaciendo ...................... 40, 53
transferir elementos del almacén
  entre máquinas .................................. 155
transformación de paquetes .................... 80

U
UEFI, cargador de arranque ......................... 396
UEFI, instalación .................................... 30
ulimit ............................................... 196
USB_ModeSwitch ................................ 202
usuarios ............................................ 176
usuarios de construcción ..................... 9
Índice de conceptos

V
valores monádicos ........................................ 104
variaciones de paquetes .......................... 122
Varnish .................................................. 310
versión de paquete ................................. 440
Vim, edición de código Scheme .................. 446
VM .......................................................... 402
VPN (red privada virtual) ...................... 335
VPS (servidor virtual privado) ................ 33
vuelta atrás ........................................... 40, 53, 402
vuelta-a-trás, del sistema operativo ....... 167
vulnerabilidades de seguridad ............. 142, 425

W
weather, disponibilidad de sustituciones .... 157
Web ...................................................... 320
web ...................................................... 303
wesnothd ............................................... 380
wicd ...................................................... 201
WiFi .................................................... 29, 201
WiFi, soporte hardware ......................... 23
WPA Supplicant .................................. 204
www ...................................................... 303

X
X11 ....................................................... 218
X11, ingreso al sistema ............................ 222
X11, tipos de sesión ................................ 218
XKB, distribuciones de teclado ........... 179
xlsfonts ............................................... 21
XMPP ................................................... 276
Xorg, configuración ................................. 222
xterm ................................................... 21

Z
zabbix zabbix-agent ............................... 293
zabbix zabbix-front-end ....................... 294
zabbix zabbix-server ......................... 291
Índice programático

#
# exp. .................................................. 111
,
, ..................................................... 78

( (gexp .................................................. 111
,
, ..................................................... 79
,0 ...................................................... 79

> == .................................................... 105

‘ ‘ ...................................................... 79

A
accountsservice-service ................................ 241
add-text-to-store ..................................... 100
agetty-service ........................................ 185
alsa-service-type ..................................... 245

B
base-initrd ............................................. 305
binary-file ............................................. 107
bluetooth-service ...................................... 244
build-derivations ..................................... 100
build-expression->derivation ......................... 103

cat-avatar-generator-service ......................... 319
close-connection ...................................... 100
colord-service-type ................................... 243
computed-file .......................................... 114
connman-service-type ................................ 203
cups-service-type ...................................... 224
current-state .......................................... 107

d
dbus-service .......................................... 240
derivation .............................................. 102
dhcppd-service-type ................................... 200
dnodoc-service .......................................... 386
directory-union ........................................ 116
dnsmasq-service-type .................................. 332
dovecot-service ........................................ 250
dropbear-service ........................................ 215

E
earlyoom-service-type .................................. 383
elogind-service ......................................... 240
enlightenment-desktop-service-type ..................... 239
exim-service-type ....................................... 271
expression->initrd ...................................... 306
extra-special-file ...................................... 184

F
file->udev-rule ......................................... 193
file-append ............................................. 116
file-system-label ...................................... 172
file-union .............................................. 116
fold-services .......................................... 418

g
geoclue-application-name ................................ 243
geoclue-service ......................................... 244
getmail-service-type ................................... 272
gexp->derivation ........................................ 113
gexp->file ............................................... 115
gexp->script ............................................ 114
gexp? ..................................................... 112
git-daemon-service ..................................... 364
git-http-nginx-location-configuration .................. 366
guix-publish-service-type ............................... 195

H
host-name-service ....................................... 184
httpd-config-file ...................................... 305
httpd-configuration .................................... 304
httpd-module ............................................ 304
httpd-service-type ..................................... 303
httpd-virtualhost ...................................... 306
<table>
<thead>
<tr>
<th>I</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>imap4d-service-type</td>
<td>nginx-configuration</td>
</tr>
<tr>
<td>inetd-service-type</td>
<td>nginx-php-fpm-location</td>
</tr>
<tr>
<td>inferior-for-channels</td>
<td>nginx-service-type</td>
</tr>
<tr>
<td>inferior-package-description</td>
<td>nsd-service</td>
</tr>
<tr>
<td>inferior-package-home-page</td>
<td>open-connection</td>
</tr>
<tr>
<td>inferior-package-inputs</td>
<td>open-inferior</td>
</tr>
<tr>
<td>inferior-package-location</td>
<td>opensmtpd-service-type</td>
</tr>
<tr>
<td>inferior-package-name</td>
<td>openssh-service-type</td>
</tr>
<tr>
<td>inferior-package-native-inputs</td>
<td>openssh-service-type</td>
</tr>
<tr>
<td>inferior-package-propagated-inputs</td>
<td>openssh-service-type</td>
</tr>
<tr>
<td>inferior-package-search-paths</td>
<td>openssh-service-type</td>
</tr>
<tr>
<td>inferior-package-synopsis</td>
<td>openvpn-client-service</td>
</tr>
<tr>
<td>inferior-package-transitive-native-search-paths</td>
<td>openvpn-client-service</td>
</tr>
<tr>
<td>inferior-package-transitive-propagated-inputs</td>
<td>openvpn-client-service</td>
</tr>
<tr>
<td>inferior-package-version</td>
<td>openvpn-server-service</td>
</tr>
<tr>
<td>inferior-package?</td>
<td>openswitch-service-type</td>
</tr>
<tr>
<td>inferior-packages</td>
<td>operating-system</td>
</tr>
<tr>
<td>inputattach-service-type</td>
<td>operating-system-derivation</td>
</tr>
<tr>
<td>interned-file</td>
<td>P</td>
</tr>
</tbody>
</table>

| K                        |                         |
| kmson-service-type       | package->cross-derivation |
| knot-resolver-service-type | package->derivation    |
| knot-service-type        | package->derivation     |

| L                       |                         |
| ladsap-service-type      | package->cross-derivation |
| libvirt-service-type     | package-cross-derivation |
| lirc-service            | package-derivation      |
| local-file              | package-file            |
| login-service           | package-input-rewriting  |
| lookup-inferior-packages | package-input-rewriting/spec |
| lookup-qemu-platforms   | package-input-rewriting  |
| lower-object            | package-input-rewriting  |
| lsh-service             | package-input-rewriting  |

| M                       |                         |
| mail-aliases-service-type | packages->manifest    |
| mate-desktop-service-type | plain-file            |
| mbegin                    | polkit-service         |
| mingetty-service         | postgresql-service     |
| mixed-text-file          | program-file           |
| mlet                      | prosody-service-type   |
| mlet*                     | pulseaudio-service-type |
| modify-services          | Q                        |
| munless                   | qemu-platform-name     |
| mvhen                     | qemu-platform?         |
| mysql-service            | quasiquote             |
|                           | quote                   |

| R                       |                         |
| raw-initrd               | R                        |
| return                   | rngd-service            |
| run-with-state           | rsync-service-type      |
| run-with-store           |                         |
Índice programático

S

sane-service-type........................................ 244
scheme-file.............................................. 115
screen-locker-service................................. 224
service..................................................... 416
service-extension....................................... 418
service-extension?....................................... 418
service-kind............................................. 416
service-value............................................ 416
service?.................................................. 416
set-current-state....................................... 107
set-xorg-configuration......................... 180, 223
simple-service............................................ 418
source-module-closure................................. 110
specification->package................................. 163
specifications->manifest............................... 39
spice-vagent-service................................. 385
state-pop................................................ 107
state-push............................................ 107
static-networking-service............................. 201
syslog-service......................................... 191

T

text-file................................................. 107
text-file*................................................ 115
this-operating-system................................. 172
this-package............................................ 83
tlp-service-type........................................ 344
tor-hidden-service...................................... 210

U

udev-rule.................................................. 192
udev-service.............................................. 192
udisks-service.......................................... 243
unquote.................................................... 79
unquote-splicing........................................ 79
uuid....................................................... 173

V

valid-path?............................................... 100
virtlog-service-type................................. 361

W

wicd-service.............................................. 201
with-extensions................................. 111, 112
with-imported-modules............................. 110, 112
with-monad.............................................. 105
with-parameters....................................... 117

X

xorg-start-command................................. 224