

Compilación del núcleo

No es muy frecuente compilar una nueva versión del núcleo a partir de los archivos fuente. Esto puede ir en contra del contrato de soporte de la distribución usada.

Sin embargo podemos querer modificar los parámetros del núcleo, añadir o quitar componentes estáticos o módulos dinámicos, para integrar nuevos tipos de dispositivos, corregir algunos errores u optimizar el rendimiento y la ocupación de la memoria.

En este caso, el procedimiento que se tiene que seguir presenta distintas etapas, la configuración del núcleo que se generará es la más delicada. La compilación efectiva puede durar algunas decenas de minutos, o incluso más tiempo según el tipo de procesador y la memoria disponible. Cuando el nuevo núcleo haya sido generado, hay que probarlo con mucho cuidado antes de considerar ponerlo en producción.

1. Descarga de las fuentes del núcleo

Los archivos fuentes del núcleo están disponibles bajo dos formas: paquetes de software o archivos comprimidos.

a. Paquete

La distribución de la máquina objetivo proporciona generalmente paquetes de software que contienen los archivos fuentes del núcleo soportado por su versión de distribución. Si el paquete corresponde a la versión que desea compilar, es el método que deberá usar preferiblemente.

Los archivos fuentes del núcleo se instalarán en uno de los directorios en `/usr/src/kernels` o en el directorio `/usr/src/linux*`, el nombre del directorio corresponderá a la versión del núcleo.

Distribuciones de tipo Red Hat

A menudo, la compilación del núcleo está vinculada con un módulo del núcleo y no con el núcleo completo.

En este caso, las distribuciones de tipo Red Hat proponen un paquete de software adaptado: `kernel-devel`.

Si necesita compilar una versión personalizada del núcleo, tendrá que usar el método por archivo comprimido.

Ejemplo

Instalación del paquete `kernel-devel` en una distribución CentOS 8:

```
yum install kernel-devel
```

```
Última comprobación de caducidad de metadatos hecha hace 0:11:07, el jue 14  
oct 2021 15:48:07 EDT.
```

```
Dependencias resueltas.
```

```
=====
=====
=====
Paquete                Arquitectura
Versión                Repositorio
Tam.
=====
=====
=====
```

```
Instalando:
```

```
kernel-devel           x86_64
  4.18.0-305.19.1.el8_4    baseos
                18 M
```

```
Resumen de la transacción
```

```
=====
=====
=====
```

```
Instalar 1 Paquete
```

```
Tamaño total de la descarga: 18 M
```

```
Tamaño instalado: 48 M
```

```
¿Está de acuerdo [s/N]?:
```

```
[...]
```

```
Instalado:
```

```
kernel-devel-4.18.0-305.19.1.el8_4.x86_64
```

La versión del núcleo corresponde a la del sistema:

```
uname -r
4.18.0-305.19.1.el8_4.x86_64
```

Los archivos están instalados en un directorio de `/usr/src/kernels` :

```
ls -l /usr/src/kernels
total 4
drwxr-xr-x. 23 root root 4096 oct 14 16:00 4.18.0-305.19.1.el8_4.x86_64
ls -l /usr/src/kernels/4.18*
total 5480
drwxr-xr-x. 26 root root 4096 oct 14 15:59 arch
drwxr-xr-x. 3 root root 78 oct 14 15:59 block
drwxr-xr-x. 2 root root 37 oct 14 15:59 certs
drwxr-xr-x. 4 root root 76 oct 14 15:59 crypto
drwxr-xr-x. 136 root root 4096 oct 14 15:59 drivers
drwxr-xr-x. 2 root root 22 oct 14 15:59 firmware
drwxr-xr-x. 73 root root 4096 oct 14 15:59 fs
drwxr-xr-x. 32 root root 4096 oct 14 16:00 include
drwxr-xr-x. 2 root root 37 oct 14 16:00 init
drwxr-xr-x. 2 root root 22 oct 14 16:00 ipc
-rw-r--r--. 1 root root 575 sep 15 11:48 Kconfig
drwxr-xr-x. 18 root root 4096 oct 14 16:00 kernel
drwxr-xr-x. 20 root root 4096 oct 14 16:00 lib
-rw-r--r--. 1 root root 61581 sep 15 11:48 Makefile
-rw-r--r--. 1 root root 1325 sep 15 11:48 Makefile.rhelver
drwxr-xr-x. 3 root root 71 oct 14 16:00 mm
-rw-r--r--. 1 root root 1307275 sep 15 11:48 Module.symvers
drwxr-xr-x. 72 root root 4096 oct 14 16:00 net
drwxr-xr-x. 27 root root 4096 oct 14 16:00 samples
drwxr-xr-x. 13 root root 8192 oct 14 16:00 scripts
drwxr-xr-x. 10 root root 176 oct 14 16:00 security
drwxr-xr-x. 26 root root 4096 oct 14 16:00 sound
-rw-r--r--. 1 root root 4166164 sep 15 11:48 System.map
drwxr-xr-x. 29 root root 4096 oct 14 16:00 tools
drwxr-xr-x. 2 root root 37 oct 14 16:00 usr
drwxr-xr-x. 4 root root 44 oct 14 16:00 virt
-rw-r--r--. 1 root root 41 sep 15 11:48 vmlinux.id
```

Distribuciones de tipo Debian

Las distribuciones de tipo Debian presentan un paquete de software que contiene las fuentes de la última versión del núcleo, incluyendo las modificaciones propias Debian: `linux-source`.

Ejemplo

Instalación del paquete `linux-source` en una distribución Debian 10:

apt-get install linux-source

Leyendo lista de paquetes... Hecho

Creando árbol de dependencias

Leyendo la información de estado... Hecho

El paquete indicado a continuación se instaló de forma automática y ya no es necesario.

gststreamer1.0-pulseaudio

Utilice «apt autoremove» para eliminarlo.

Se instalarán los siguientes paquetes adicionales:

bc bison flex libbison-dev libfl-dev libfl2 libsigsegv2 linux-config-4.19

linux-source-4.19 m4

Paquetes sugeridos:

bison-doc flex-doc libqt4-dev pkg-config m4-doc

Se instalarán los siguientes paquetes NUEVOS:

bc bison flex libbison-dev libfl-dev libfl2 libsigsegv2 linux-config-4.19

linux-source linux-source-4.19 m4

0 actualizados, 11 nuevos se instalarán, 0 para eliminar y 7 no actualizados.

Se necesita descargar 110 MB de archivos.

Se utilizarán 113 MB de espacio de disco adicional después de esta operación.

¿Desea continuar? [S/n]

[...]

Configurando flex (2.6.4-6.2) ...

Configurando libfl-dev:amd64 (2.6.4-6.2) ...

Procesando disparadores para libc-bin (2.28-10) ...

Procesando disparadores para man-db (2.8.5-2) ...

Procesando disparadores para menu (2.1.47+b1) ...

La versión del micro corresponde a la del sistema:

```
uname -r
4.19.0-17-amd64
```

Varios archivos se instalan en los directorios de `/usr/src`. Los archivos comprimidos de las fuentes y de los parches se encuentran en `/usr/src`:

```
ls -dl /usr/src/lin*
drwxr-xr-x 2 root root 4096 oct 14 22:13 /usr/src/linux-config-4.19
drwxr-xr-x 4 root root 4096 jul 24 10:26 /usr/src/linux-headers-4.19.0-17-amd64
drwxr-xr-x 4 root root 4096 jul 24 10:26 /usr/src/linux-headers-4.19.0-17-common
lrwxrwxrwx 1 root root 24 jul 18 08:52 /usr/src/linux-kbuild-4.19 -> ../lib/
linux-kbuild-4.19
-rw-r--r-- 1 root root 157152 sep 29 20:53 /usr/src/linux-patch-4.19-rt.patch.xz
-rw-r--r-- 1 root root 106425584 sep 29 20:53 /usr/src/linux-source-4.19.tar.xz
```

b. Archivo comprimido

El sitio www.kernel.org proporciona distintas versiones del núcleo, pudiendo ser descargadas gratuitamente, bajo la forma de archivos comprimidos.

Cuando el archivo se haya descargado, hay que posicionarse en el directorio de instalación de las fuentes (generalmente `/usr/src`) y descomprimir el archivo.

Ejemplo

Se quiere comprobar una versión del núcleo con soporte a largo plazo, cercana a la versión instalada en un servidor Debian 10.

Comprobamos la versión del núcleo actual:

```
uname -r
4.19.0-17-amd64
```

En la página web <https://www.kernel.org/>, buscamos una versión longterm 4.19.x:

mainline:	5.15-rc5	2021-10-11	[tarball]		[patch]	[in
stable:	5.14.12	2021-10-13	[tarball]	[pgp]	[patch]	[in
stable:	5.13.19 [EOL]	2021-09-18	[tarball]	[pgp]	[patch]	[in
longterm:	5.10.73	2021-10-13	[tarball]	[pgp]	[patch]	[in
longterm:	5.4.153	2021-10-13	[tarball]	[pgp]	[patch]	[in
longterm:	4.19.211	2021-10-13	[tarball]	[pgp]	[patch]	[in
longterm:	4.14.250	2021-10-09	[tarball]	[pgp]	[patch]	[in
longterm:	4.9.286	2021-10-09	[tarball]	[pgp]	[patch]	[in
longterm:	4.4.288	2021-10-09	[tarball]	[pgp]	[patch]	[in
linux- next:	next-20211013	2021-10-13				

La versión 4.19.211 es la más cercana a la versión instalada.

Se descarga el archivo comprimido (tarball):

```
wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.19.211.tar.xz
--2021-10-14 22:21:19-- https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-
4.19.211.tar.xz
Resolviendo cdn.kernel.org (cdn.kernel.org)... 151.101.121.176, 2a04:4e42:1d::432
Conectando con cdn.kernel.org (cdn.kernel.org)[151.101.121.176]:443... conectado.
```

Petición HTTP enviada, esperando respuesta... 200 OK

Longitud: 103207100 (98M) [application/x-xz]

Grabando a: "linux-4.19.211.tar.xz"

linux-4.19.211.tar.xz

```
100%[=====
=====>] 98,43M 5,95MB/s en 16s
```

2021-10-14 22:21:36 (6,02 MB/s) - "linux-4.19.211.tar.xz" guardado
[103207100/103207100]

ls -l linux*

```
-rw-r--r-- 1 root root 103207100 oct 13 10:23 linux-4.19.211.tar.xz
```

El archivo está comprimido en formato `xz`.

Con la cuenta de superusuario, copiamos el archivo en `/usr/src` y, ubicándonos en este directorio, restauramos el archivo (opción `J` de `tar` para la descompresión de un formato `xj`):

cp linux-4.19.211.tar.xz /usr/src

cd /usr/src

tar xJf linux-4.19.211.tar.xz

ls -ld linux-4.19.211*

```
drwxrwxr-x 25 root root 4096 oct 13 10:10 linux-4.19.211
```

```
-rw-r--r-- 1 root root 103207100 oct 14 23:01 linux-4.19.211.tar.xz
```

El archivo ha sido restaurando creando el directorio `linux-4.19.211`:

ls -l linux-4.19.211

```
total 748
```

```
drwxrwxr-x 26 root root 4096 oct 13 10:10 arch
```

```
drwxrwxr-x 3 root root 4096 oct 13 10:10 block
```

```
drwxrwxr-x 2 root root 4096 oct 13 10:10 certs
```

```
-rw-rw-r-- 1 root root 423 oct 13 10:10 COPYING
```

```
-rw-rw-r-- 1 root root 98741 oct 13 10:10 CREDITS
```

```

drwxrwxr-x  4 root root  4096 oct 13 10:10 crypto
drwxrwxr-x 120 root root 12288 oct 13 10:10 Documentation
drwxrwxr-x 137 root root  4096 oct 13 10:10 drivers
drwxrwxr-x  2 root root  4096 oct 13 10:10 firmware
drwxrwxr-x  73 root root  4096 oct 13 10:10 fs
drwxrwxr-x  27 root root  4096 oct 13 10:10 include
drwxrwxr-x  2 root root  4096 oct 13 10:10 init
drwxrwxr-x  2 root root  4096 oct 13 10:10 ipc
-rw-rw-r--  1 root root  2245 oct 13 10:10 Kbuild
-rw-rw-r--  1 root root   563 oct 13 10:10 Kconfig
drwxrwxr-x  18 root root  4096 oct 13 10:10 kernel
drwxrwxr-x  13 root root 12288 oct 13 10:10 lib
drwxrwxr-x  5 root root  4096 oct 13 10:10 LICENSES
-rw-rw-r--  1 root root 471159 oct 13 10:10 MAINTAINERS
-rw-rw-r--  1 root root 60600 oct 13 10:10 Makefile
drwxrwxr-x  3 root root  4096 oct 13 10:10 mm
drwxrwxr-x  70 root root  4096 oct 13 10:10 net
-rw-rw-r--  1 root root   800 oct 13 10:10 README
drwxrwxr-x  27 root root  4096 oct 13 10:10 samples
drwxrwxr-x  14 root root  4096 oct 13 10:10 scripts
drwxrwxr-x  10 root root  4096 oct 13 10:10 security
drwxrwxr-x  26 root root  4096 oct 13 10:10 sound
drwxrwxr-x  32 root root  4096 oct 13 10:10 tools
drwxrwxr-x  2 root root  4096 oct 13 10:10 usr
drwxrwxr-x  4 root root  4096 oct 13 10:10 virt

```

2. Configuración y compilación del núcleo

La fase de configuración tiene como objetivo definir los componentes y los parámetros del nuevo núcleo, creando a partir de ellos un archivo `.config` que será usado para la compilación.

a. Limpieza del directorio de compilación

Si ya se han hecho otras compilaciones en este directorio, puede ser necesario eliminar los archivos generados en estas operaciones. Existen diferentes niveles de limpieza

efectuados por el comando `make`:

```
make clean
```

Limpieza mínima. Supresión de la mayoría de los archivos generados, pero se conserva la selección de la configuración y los elementos para los módulos externos.

```
make mrproper
```

Limpieza más completa. Supresión de la mayoría de los archivos generados, del archivo de configuración y de los distintos elementos guardados.

```
make distclean
```

Limpieza completa. Además de las supresiones del nivel `mrproper`, suprime el archivo de respaldo del editor y los archivos de parches (*patches*).

b. Generación de un archivo de respuesta

La configuración de la compilación está determinada por el contenido del archivo de texto `.config`, creado en el directorio de compilación.

El archivo `.config` permite especificar, para cada elemento del núcleo, si tiene que estar integrado directamente en el núcleo, generado bajo la forma de un módulo dinámico LKM, o si no debe ser usado, así como los valores de los diferentes parámetros del núcleo.

Varios comandos, en modo de carácter, semigráfico o gráfico, permiten gestionar ese archivo:

<code>make config</code>	Modo en línea de comandos: una solicitud por cada elemento.
<code>make menuconfig</code>	Modo semi-gráfico: menú de configuración de los elementos.
<code>make xconfig</code>	Interfaz gráfica de tipo X Window.
<code>make gconfig</code>	Interfaz gráfica de tipo Gnome.
<code>make defconfig</code>	Crea un archivo <code>.config</code> por defecto.
<code>make oldconfig</code>	Genera un archivo <code>.config</code> a partir del una versión del núcleo precedente.

Esta fase es particularmente delicada, hay varias centenas de elementos configurables.



Por defecto, el archivo de configuración del núcleo actual sirve de base del comando `make menuconfig` para el nuevo archivo de configuración. Este archivo se encuentra generalmente en el directorio de arranque `/boot`.

Ejemplo

En una distribución Debian 10 para una plataforma AMD 64 bits:

```
ls -l /boot/config*
```

```
-rw-r--r-- 1 root root 206267 sep 29 20:53 /boot/config-4.19.0-18-amd64
```

Este archivo de texto corresponde al archivo `.config` del núcleo `4.19.0` instalado.

c. Ejemplo de configuración

En este ejemplo utilizamos el comando en modo semi-gráfico, `make menuconfig`. Se efectúan una o distintas modificaciones, por ejemplo desactivando el soporte de los sistemas de archivos de tipo ReiserFS.

Nos posicionamos en el directorio de compilación del nuevo núcleo:

```
cd /usr/src/linux-4.19.211
```

Lanzamos el comando `make menuconfig`:

```
make menuconfig
```

Mediante la opción de menú `File systems --->`, desactivamos el soporte del tipo de sistema de archivos ReiserFS:

```
<M> Reiserfs support
```

Reemplazado por:

```
< > Reiserfs support
```

Salimos del menú, guardando las modificaciones en el archivo `.config`.

Comparamos el archivo creado con el del núcleo actual:

```
diff .config /boot/config-4.19.0-17-amd64 | wc -l
46
```

Las modificaciones son solo conciernen a ReiserFS.

d. Compilación del núcleo

Para poder compilar el núcleo, es necesario haber instalado diferentes paquetes de desarrollo.

La compilación del núcleo se hace generalmente usando el comando `make bzImage`.

Esta fase puede durar mucho tiempo, de algunas decenas de minutos hasta algunas horas.



Para algunas versiones del núcleo y de arquitectura de software, utilizamos el comando `makezImage`, que crea una imagen comprimida usando un tipo de compresión más antiguo, a condición de que el tamaño del núcleo descomprimido sea inferior a 512 KB.

Ejemplo

Compilación del núcleo configurado anteriormente.

Para evitar errores durante la compilación con esta versión del núcleo, habrá que modificar el archivo `.config` posicionando los parámetros siguientes:

```
cd /usr/src/linux-4.19.211
vi .config
CONFIG_SYSTEM_TRUSTED_KEYS=""
CONFIG_RYPOLINE=y
```

Lanzamos la compilación en segundo plano:

```
nohup make bzImage > bz.log 2>&1 &
```

Alrededor de una hora después, el tratamiento se termina:

```
vi bz.log
nohup: no se tendrá en cuenta la entrada
CALL scripts/checksyscalls.sh
DESCEND objtool
HOSTCC scripts/sign-file
HOSTCC scripts/extract-cert
CHK include/generated/compile.h
UPD include/generated/compile.h
CC init/main.o
CC init/version.o
CC init/do_mounts.o
[...]
Setup is 17212 bytes (padded to 17408 bytes).
System is 5141 kB
CRC a64b529f
Kernel: arch/x86/boot/bzImage is ready (#1)
```

Los elementos compilados se encuentran en el subdirectorio `arch/x86/boot`:

```
ls arch/x86/boot
a20.c      cpucheck.c      install.sh      regs.c      video-bios.o
a20.o      cpucheck.o      main.c         regs.o      video.c
apm.c      cpuflags.c      main.o         setup.bin   video.h
bioscall.o cpuflags.h      Makefile       setup.elf   video-mode.c
bioscall.S cpuflags.o      memory.c       setup.ld    video-mode.o
bitops.h   cpu.o           memory.o       string.c    video.o
boot.h     cpustr.h        mkcpustr       string.h    video-vesa.c
bzImage    ctype.h         mkcpustr.c     string.o    video-vesa.o
cmdline.c  early_serial_console.c mtools.conf.in tools       video-vga.c
cmdline.o  early_serial_console.o pm.c          tty.c       video-vga.o
code16gcc.h edd.c          pmjump.o      tty.o       vmlinux.bin
compressed edd.o          pmjump.S      version.c   voffset.h
copy.o     genimage.sh    pm.o          version.o   zoffset.h
```

```
copy.S    header.o    printf.c   vesa.h
cpu.c     header.S    printf.o   video-bios.c
```

El archivo imagen es `arch/x86/boot/bzImage` :

```
ls -l arch/x86/boot/bzImage
-rw-r--r-- 1 root root 5281664 oct 17 21:25 arch/x86/boot/bzImage
file arch/x86/boot/bzImage
arch/x86/boot/bzImage: Linux kernel x86 boot executable bzImage, version
4.19.211 (root@debian10) #1 SMP Sun Oct 17 20:58:29 CEST 2021, RO-rootFS,
swap_dev 0x5, Normal VGA
```

e. Compilación de los módulos del núcleo

La etapa siguiente consiste en compilar los módulos externos del núcleo, usando el comando `make modules`.

Este comando puede durar mucho tiempo, incluso algunas horas.

Ejemplo

Compilación de los módulos del núcleo configurado anteriormente:

```
cd /usr/src/linux-4.19.211
nohup make modules > mod.log 2>&1 &
vi mod.log
nohup: no se tendrá en cuenta la entrada
CALL scripts/checksyscalls.sh
DESCEND objtool
CC [M] arch/x86/crypto/glue_helper.o
AS [M] arch/x86/crypto/aes-x86_64-asm_64.o
CC [M] arch/x86/crypto/aes_glue.o
LD [M] arch/x86/crypto/aes-x86_64.o
AS [M] arch/x86/crypto/des3_edc-asm_64.o
[...]
LD [M] sound/xen/snd_xen_front.ko
```

```
CC virt/lib/irqbypass.mod.o
LD [M] virt/lib/irqbypass.ko
```

La generación se ha terminado, después de algunas horas de tratamiento.

f. Gestión de los módulos del núcleo por DKMS

El paquete de software DKMS (*Dynamic Kernel Module Support*) ofrece un conjunto de herramientas que permiten facilitar la generación de los módulos LKM en caso de implementación de una nueva versión del núcleo.

Las fuentes de los módulos existentes se pueden poner en formato DKMS gracias al comando `dkms`. También se pueden instalar los módulos proporcionados en formato DKMS en paquetes de software (cuyo nombre generalmente termina por `dkms`).

El comando `dkms` y sus numerosos subcomandos permiten listar los módulos existentes, y volver a crearlos para la versión actual del núcleo o para una nueva versión.

3. Instalación del nuevo núcleo

Una vez que se haya generado el archivo imagen del núcleo y los módulos, hay que instalar el conjunto en diferentes directorios del sistema, generar un archivo de tipo disco virtual (*RAM DISK*) `initramfs`, y configurar el gestor de arranque para que proponga en su menú de arranque la carga del nuevo núcleo.

a. Instalación de los módulos

El comando `make modules_install` realiza la instalación de los módulos generados. Son copiados en el directorio `/lib/modules`, en un subdirectorio que corresponde a la versión del núcleo.

Ejemplo

Se instalan los módulos anteriormente generados:

```

cd /usr/src/linux-4.19.211
nohup make modules_install > modinst.log 2>&1 &
vi modinst.log
nohup: no se tendrá en cuenta la entrada
INSTALL arch/x86/crypto/aegis128-aesni.ko
INSTALL arch/x86/crypto/aegis128l-aesni.ko
INSTALL arch/x86/crypto/aegis256-aesni.ko
INSTALL arch/x86/crypto/aes-x86_64.ko
INSTALL arch/x86/crypto/aesni-intel.ko
[...]
INSTALL sound/xen/snd_xen_front.ko
INSTALL virt/lib/irqbypass.ko
DEPMOD 4.19.211

```

Después de algunos minutos, los módulos se encuentran instalados.

```

ls /lib/modules/4.19.211
build      modules.alias.bin  modules.dep  modules.order  modules.symbols.bin
kernel     modules.builtin     modules.dep.bin  modules.softdep  source
modules.alias  modules.builtin.bin  modules.devname  modules.symbols

```

b. Instalación del núcleo

Para que el archivo imagen del núcleo pueda ser cargado durante el arranque del sistema, hay que copiarlo en el directorio `/boot`. El nombre del archivo destino es libre, pero generalmente se usa la forma `vmlinuz-X.Y.Z-IdArch` donde `X.Y.Z` corresponde a la versión del núcleo y `IdArch` al tipo de arquitectura del procesador.



También se puede copiar el archivo `.config` en un archivo `config-X.Y.Z-IdArch` para conservar la configuración del núcleo generado.

Ejemplo

Instalación del archivo imagen generado anteriormente:

```
cp arch/x86/boot/bzImage /boot/vmlinuz-4.19.211-amd64
```

c. Creación del archivo de disco virtual de los módulos

En el momento de la carga, el núcleo necesita un cierto número de módulos. Para ello, creamos un archivo de disco virtual que contenga esos módulos, archivo que será cargado en memoria como un sistema de archivos virtuales.

Se pueden usar dos comandos para crear este archivo, según el tipo de archivo de disco virtual deseado.

```
mkinitrd ArchivImg Versión
```

Este comando es el más antiguo. Genera un archivo de tipo `initrd`. Tiende poco a poco a convertirse en obsoleto en las versiones recientes de las distribuciones.

```
mkinitramfs [ -c TipoComp ] -o ArchivImg Versión
```

Este comando genera un archivo de tipo `initramfs`. Hace una llamada automática al comando `dracut`.

El argumento `ArchivImg` corresponde al camino de acceso completo del archivo imagen que se va a crear. El argumento `Versión` corresponde a la versión del núcleo que se va a instalar y al directorio correspondiente en el directorio de los módulos `/lib/modules`.

La opción `-c TipoComp` permite especificar el tipo de compresión `TipoComp`; por defecto se usa el formato `gzip`.

El archivo de configuración `/etc/initramfs-tools/initramfs.conf` permite configurar el comando, en particular la elección de los módulos que se incluirán estará

determinada por la línea:

```
MODULES=[most | dep]
```

El valor `dep` limita el número de módulos que se incluirán y el tamaño del archivo de disco virtual; por defecto es `most`, es decir, el máximo de módulos.



Las versiones recientes de las distribuciones de tipo Red Hat usan el comando `mkinitrd` en lugar del comando `mkinitramfs`, pero genera un archivo de disco virtual de tipo `initramfs`.

Ejemplo

Creación del archivo de disco virtual para los módulos del núcleo generado anteriormente.

```
mkinitramfs -c xz -o initrd.img-4.19.211-amd64 4.19.211  
[...]
```

El archivo ha sido generado:

```
ls -l initrd.img-4.19.211-amd64  
-rw-r--r-- 1 root root 15670476 oct 23 19:51 initrd.img-4.19.211-amd64  
file initrd.img-4.19.211-amd64  
initrd.img-4.19.211-amd64: XZ compressed data
```

Está comprimido en formato `xz`.

Lo copiamos en `/boot`:

```
cp initrd.img-4.19.211-amd64 /boot
```

d. Configuración del gestor de arranque

La última etapa consiste en modificar el archivo de configuración del gestor de arranque para añadir la posibilidad de cargar el nuevo núcleo. El procedimiento que se tiene que seguir depende del tipo de gestor de arranque utilizado (ver el capítulo relativo al arranque del sistema).



Es prudente asegurarse de que el núcleo actual sigue siendo el núcleo cargado por defecto, hasta que el nuevo núcleo no haya sido completamente comprobado y validado.

Ejemplo

El gestor de arranque GRUB 2 proporciona un comando que actualiza automáticamente el menú de arranque con los nuevos archivos de imágenes de núcleo detectados en el directorio `/boot`:

update-grub

Generando un fichero de configuración de grub...

Found background image: /usr/share/images/desktop-base/desktop-grub.png

Encontrada imagen de linux: /boot/vmlinuz-4.19.211-amd64

Encontrada imagen de memoria: /boot/initrd.img-4.19.211-amd64

Encontrada imagen de linux: /boot/vmlinuz-4.19.0-8-amd64

Encontrada imagen de memoria: /boot/initrd.img-4.19.0-8-amd64

hecho

El comando ha encontrado la nueva imagen del núcleo y lo ha añadido al menú de arranque, como núcleo por defecto.

Se reinicia el sistema:

shutdown -r 0

El sistema se reinicia cargando la nueva versión del núcleo. Sin embargo, la antigua versión del núcleo sigue siendo propuesta en la elección del menú de arranque **Opciones avanzadas para Debian**.

Después del reinicio, comprobamos la versión del núcleo cargado:

```
uname -r 4.19.211
```

El nuevo núcleo ha sido cargado. Ahora solamente queda comprobarlo.