Compilación del kernel

Desde el punto de vista de la compilación, el kernel (también llamado núcleo) es casi como una aplicación más, con su código fuente, un procedimiento de compilación y un procedimiento de instalación.

1. Los componentes del kernel

El kernel Linux es el responsable de la gestión del hardware. El concepto de controlador de dispositivo no existe directamente en entornos Linux debido a que los elementos que permiten comunicarse correctamente con un periférico están incluidos en el código del kernel. Uno se acostumbra muy rápido a la comodidad de esta situación: el kernel recién incluido en una distribución Linux puede gestionar directamente el conjunto de dispositivos de un sistema sin tener que instalar controladores adicionales. Por el contrario, el código del kernel tiene la tendencia cada vez más a ser más pesado para gestionar el conjunto de dispositivos existente y su carga total provocaría un consumo de memoria excesivo. Por esta razón, el kernel tiene una estructura modular y solo se cargan en memoria los módulos necesarios para el correcto funcionamiento del sistema.

a. El corazón del kernel

Se puede llamar «corazón del kernel» a la parte irreductible del kernel, que es la que se carga en su totalidad en memoria. Solo contiene elementos de los que se está seguro que se necesitarán. El corazón del kernel es un archivo que se encuentra en el directorio /boot y cuyo tamaño es de algunos MB.

b. Módulos

La importancia de los módulos del kernel

Los módulos tienen un papel primordial ya que muchas de las funciones básicas se gestionan en forma de módulos. Si un kernel no dispone de los módulos necesarios para el funcionamiento del sistema, las funciones simplemente no estarán disponibles.

Intento de carga de un recurso no soportado

Este ejemplo se realiza en un sistema cuyo kernel no soporta el formato de sistema de archivos ext3.

```
light:/mnt# mount /dev/hda3 partition
mount: unknown filesystem type 'ext3'
light:/mnt#
```

Los módulos son archivos con la extensión **.ko** que se cargan en memoria en función de las necesidades. Existe a nuestra disposición una serie de comandos que permiten listar los módulos cargados, retirarlos de memoria o cargar otros nuevos.


Los kernels de versiones antiguas (2.4 especialmente) usan archivos de módulos con la extensión «.o».

Operaciones puntuales con módulos

Visualización de módulos cargados en memoria

```
lsmod
```

Visualización de los módulos disponibles en el sistema

```
modprobe -l
```

Los archivos que corresponden a estos módulos se encuentran normalmente en el directorio **/lib/modules**, dentro de una estructura de directorios cuyo directorio principal es el nombre del kernel actual, tal y como se obtiene con el comando **uname -r**.

Quitar un módulo cargado en memoria

```
rmmod nombre_módulo
```

o

```
modprobe -r nombre_módulo
```

Donde *nombre_módulo* representa el nombre del módulo presente en memoria tal y como se muestra con el comando **lsmod**. Ambos comandos (**rmmod** y **modprobe -r**) producen el mismo resultado.

Carga de un módulo en memoria

```
insmod archivo_módulo
```

o

```
modprobe nombre_módulo
```

Donde *nombre_módulo* representa el nombre del módulo tal y como será mostrado por el comando **lsmod** y *archivo_módulo* representa el nombre del archivo de módulo presente en disco. De hecho, el nombre del módulo se obtiene retirando la extensión **.ko** al nombre del archivo.

Carga de un módulo

La carga manual del módulo que faltaba anteriormente hace posible que se pueda montar la partición ext3.

```
light:/mnt# insmod /lib/modules/2.6.26-2-686/kernel/fs/ext3/ext3.ko
light:/mnt# mount /dev/hda3 partition
light:/mnt# mount
/dev/hda1 on / type ext2 (rw,errors=remount-ro)
(...)
/dev/hda3 on /mnt/partition type ext3 (rw)
light:/mnt#
```

Carga forzada de un módulo

Los módulos se cargan en principio durante el arranque en función del hardware que se haya detectado. Sin embargo, se puede forzar la carga de un módulo rellenando un archivo de configuración de módulos. Cualquier módulo mencionado en el archivo **/etc/modules** se cargará por defecto en el arranque.

Configuración de módulos

El archivo **/etc/modules.conf** permite configurar algunos módulos y especialmente definir asociaciones forzadas entre dispositivos y módulos.

Ejemplo de archivo /etc/modules.conf

```
# Asociación forzada del controlador tg3 con la tarjeta de red
alias eth0 tg3
```

A título de comprobación, o para ver si las asociaciones entre el hardware y los módulos han sido realizadas correctamente, se puede mostrar información acerca de los módulos cargados con el comando **modinfo**.

Visualización de la información relacionada con un módulo

Se ve especialmente el archivo .ko que contiene el código del módulo, alguna información relativa al entorno y los alias gestionados dinámicamente por el sistema para el hardware ligado a este módulo.

```
root@servidor:/boot$ modinfo r8169
filename:       /lib/modules/2.6.32-24-generic/kernel/drivers/net/r8169.ko
version:        2.3LK-NAPI
license:        GPL
description:    RealTek RTL-8169 Gigabit Ethernet driver
author:         Realtek and the Linux r8169 crew <netdev@vger.kernel.org>
srcversion:     D37E06388C6313C1D062CC3
alias:          pci:v00000001d00008168sv*sd00002410bc*sc*i*
alias:          pci:v00001737d00001032sv*sd00000024bc*sc*i*
alias:          pci:v000016ECd00000116sv*sd*bc*sc*i*
alias:          pci:v00001259d0000C107sv*sd*bc*sc*i*
alias:          pci:v00001186d00004300sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008169sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008168sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008167sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008136sv*sd*bc*sc*i*
alias:          pci:v000010ECd00008129sv*sd*bc*sc*i*
depends:        mii
vermagic:       2.6.32-24-generic SMP mod_unload modversions
parm:           rx_copybreak:Copy breakpoint for copy-only-tiny-frames (int)
parm:           use_dac:Enable PCI DAC. Unsafe on 32 bit PCI slot. (int)
parm:           debug:Debug verbosity level (0=none, ..., 16=all) (int)
root@servidor:/boot$
```

c. Alrededor del kernel

Ya se ha explicado que el kernel se constituye de una entidad indivisible y de módulos cargados en memoria bajo demanda. En la fase de arranque, el gestor de arranque carga el kernel, así como los módulos correspondientes a la configuración de hardware del sistema. Para acelerar la fase de detección del hardware y la carga de módulos asociada, la mayoría de los sistemas modernos usan un ramdisk (disco virtual cuyo soporte físico es la memoria principal) que contiene el conjunto de módulos. Este ramdisk se genera una vez se ha compilado el kernel y se llama directamente por el gestor de arranque.

d. Gestión de versiones del kernel

El kernel lleva un número de versión de tipo A.B.C, por ejemplo 2.6.15. «A» determina la versión principal del kernel: hasta hace poco la versión 2. «B» representa la versión actual del kernel. Este valor es sistemáticamente par en las versiones estables e impar en las versiones de desarrollo. Finalmente, «C» se incrementa en función de las evoluciones menores del kernel, básicamente correcciones de errores y actualizaciones de nuevo hardware.

En julio de 2011, el núcleo alcanzó la versión 3.0 teniendo como cambio principal, según su creador, Linus Torvalds: «nada, absolutamente nada». Este cambio se recibió como una especie de capricho, más o menos para celebrar los veinte años del núcleo de Linux. La evolución de esta versión, la más espectacular, para el gran público ha sido el soporte a los periféricos Kinect de Microsoft. Una adición entre tantas otras, soporte de nuevo hardware, correcciones varias y otras evoluciones menores.

Las distribuciones actuales no buscan especialmente proporcionar la última versión disponible del núcleo. El número de funciones implementadas por el núcleo hace que la validación de una versión antes de su distribución sea difícil y compleja. Los sistemas Linux orientados al gran público son más bien vanguardistas e intentan proporcionar las últimas versiones del núcleo, mientras que las destinadas al mundo empresarial premian la estabilidad y liberan núcleos en versiones más antiguas, que de algún modo han superado las pruebas de estabilidad. En junio de 2012, Red Hat ofrecía un núcleo en versión 2.6.32, Ubuntu 3.2.0, mientras que el último núcleo estable disponible en el sitio web de kernel.org estaba en versión 3.4.4.

El comando **uname -r** permite visualizar la versión del kernel en ejecución.

Visualización de la versión del kernel actual

```
toto@servidor:~$ uname -r
2.6.32-24-generic
toto@servidor:~$
```

2. Procedimiento de compilación y de utilización

El procedimiento de compilación siempre debe consultarse en el archivo **README** presente con las fuentes del kernel. Los elementos específicos del kernel están documentados en el directorio **Documentation** proporcionado con las fuentes. El archivo **README** solo documenta el procedimiento de compilación.

a. Obtención de fuentes

El código fuente del kernel se puede descargar de forma gratuita desde el sitio web http://www.kernel.org. Las principales versiones están disponibles. Los enlaces «Full source» permiten descargar el código fuente completo del kernel.

El kernel se libera en forma de archivo tar.bz2, por lo que primeramente hay que descomprimirlo. Como para cualquier compilación de aplicación, la mayor parte del trabajo se realizará en el directorio creado en la extracción del archivo.

Si se trabaja sobre las fuentes del kernel copiadas en la instalación del sistema, el directorio de trabajo debería ser /usr/src/linux. Naturalmente, la documentación se encontrará entonces en /usr/src/linux/Documentation. En caso de trabajar con fuentes nuevas, se recomienda utilizar un directorio neutro.

b. Generación del archivo de configuración

La compilación se realiza en función de la información albergada en el archivo **.config** que se encuentra en la raíz del directorio de fuentes. Este archivo indica para cada elemento del kernel si debe estar presente en el corazón del kernel, presente en forma de módulo o ausente del kernel compilado.

Según el sistema usado, hay varios medios a nuestra disposición para generar este archivo de configuración.

Generación del archivo de configuración: comandos posibles	
make config	Va realizando preguntas al usuario para cada uno de los módulos.
make menuconfig	Presenta una interfaz de texto mejorada.
make xconfig	Presenta una interfaz gráfica.
make gconfig	Presenta una interfaz gráfica.
make defconfig	Genera un archivo de configuración basándose en todos los valores de compilación por defecto.
make oldconfig	Genera un archivo de configuración basándose en un archivo .config ya utilizado para una versión más antigua del kernel.

Aunque la compilación del kernel no presenta ninguna dificultad particular, informar el archivo de configuración requiere una gran capacidad y el conocimiento preciso del hardware.

Ejemplo de creación del archivo de configuración

La compilación detallada del núcleo requiere el conocimiento de todas las tecnologías de hardware gestionadas por este núcleo.

```
[root@beta linux-2.6.34.4]# make config
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/basic/docproc
  HOSTCC  scripts/basic/hash
  HOSTCC  scripts/kconfig/conf.o
```

```
(...)
PentiumPro memory ordering errata workaround (X86_PPRO_FENCE) [Y/n/?] y
HPET Timer Support (HPET_TIMER) [Y/n/?] y
Maximum number of CPUs (NR_CPUS) [8] (NEW) 8
SMT (Hyperthreading) scheduler support (SCHED_SMT) [Y/n/?] y
Multi-core scheduler support (SCHED_MC) [Y/n/?] y
Preemption Model
  1. No Forced Preemption (Server) (PREEMPT_NONE)
> 2. Voluntary Kernel Preemption (Desktop) (PREEMPT_VOLUNTARY)
  3. Preemptible Kernel (Low-Latency Desktop) (PREEMPT)
choice[1-3]: 2
Reroute for broken boot IRQs (X86_REROUTE_FOR_BROKEN_BOOT_IRQS) [N/y/?] (NEW) n
Machine Check / overheating reporting (X86_MCE) [Y/n/?] n
Toshiba Laptop support (TOSHIBA) [M/n/y/?] n
Dell laptop support (I8K) [M/n/y/?] m
Enable X86 board specific fixups for reboot (X86_REBOOTFIXUPS) [N/y/?] n
(...)
CRC-CCITT functions (CRC_CCITT) [M/y/?] m
CRC16 functions (CRC16) [M/y/?] m
CRC calculation for the T10 Data Integrity Field (CRC_T10DIF) [N/m/y/?] (NEW) m
CRC ITU-T V.41 functions (CRC_ITU_T) [M/y/?] m
CRC32 functions (CRC32) [Y/?] y
CRC7 functions (CRC7) [N/m/y/?] (NEW) n
CRC32c (Castagnoli, et al) Cyclic Redundancy-Check (LIBCRC32C) [Y/m/?] y
#
# configuration written to .config
#


[root@beta linux-2.6.34.4]#
```

Primeras líneas del archivo de configuración

```
[root@beta linux-2.6.34.4]# head -15 .config
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.34.4
# Mon Aug 8 13:21:04 2011
#
# CONFIG_64BIT is not set
CONFIG_X86_32=y
# CONFIG_X86_64 is not set
CONFIG_X86=y
CONFIG_OUTPUT_FORMAT="elf32-i386"
CONFIG_ARCH_DEFCONFIG="arch/x86/configs/i386_defconfig"
CONFIG_GENERIC_TIME=y
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_CLOCKSOURCE_WATCHDOG=y
CONFIG_GENERIC_CLOCKEVENTS=y
[root@beta linux-2.6.34.4]#
```

Tamaño indicativo del archivo de configuración

```
[root@beta linux-2.6.34.4]# wc -l .config
3641 .config
[root@beta linux-2.6.34.4]#
```

La configuración de los módulos para una versión de kernel instalada se encuentra en el archivo config-*versión* en el directorio /boot.

Visualización de los archivos de configuración de los kernels

```
root@servidor:/boot$ ls config*
config-2.6.27-11-generic  config-2.6.32-21-generic  config-2.6.32-24-generic
config-2.6.28-16-generic  config-2.6.32-22-generic
config-2.6.31-21-generic  config-2.6.32-23-generic
root@servidor:/boot$ cat config-2.6.32-24-generic
#
# Automatically generated make config: don't edit
# Linux kernel version: 2.6.32-24-generic
# Thu Aug 19 01:38:31 2010
#
CONFIG_64BIT=y
# CONFIG_X86_32 is not set
CONFIG_X86_64=y
CONFIG_X86=y
CONFIG_OUTPUT_FORMAT="elf64-x86-64"
CONFIG_ARCH_DEFCONFIG="arch/x86/configs/x86_64_defconfig"
CONFIG_GENERIC_TIME=y
CONFIG_GENERIC_CMOS_UPDATE=y
CONFIG_CLOCKSOURCE_WATCHDOG=y
(...)
root@servidor:/boot$
```

c. Compilación del kernel y de los módulos

La compilación se realiza de la forma más trivial, simplemente tecleando el comando **make** desde el directorio raíz de las fuentes. La duración de la operación depende de la potencia de la máquina en la que se realiza, pero una buena hora es a menudo necesaria. Para un kernel con versión 2.6, el comando **make** provoca la compilación del kernel y sus módulos.

Compilación del kernel y de los módulos

Tardará alrededor de una hora o dos...

```
[root@beta linux-2.6.34.4]# make
scripts/kconfig/conf -s arch/x86/Kconfig
  CHK     include/linux/version.h
  UPD     include/linux/version.h
  CHK     include/generated/utsrelease.h
  UPD     include/generated/utsrelease.h
  CC      kernel/bounds.s
  GEN     include/generated/bounds.h
  CC      arch/x86/kernel/asm-offsets.s
(...)
  CC      arch/x86/kernel/cpu/cpufreq/speedstep-lib.o
  CC      arch/x86/kernel/cpu/cpufreq/speedstep-smi.o
  LD      arch/x86/kernel/cpu/cpufreq/built-in.o
```

```
  CC [M]  arch/x86/kernel/cpu/cpufreq/powernow-k8.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/acpi-cpufreq.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/speedstep-centrino.o
  CC [M]  arch/x86/kernel/cpu/cpufreq/p4-clockmod.o
  CC      arch/x86/kernel/cpu/mcheck/mce.o
  CC      arch/x86/kernel/cpu/mcheck/mce-severity.o
  CC      arch/x86/kernel/cpu/mcheck/mce_intel.o
(...)
```

La ejecución del comando **make** provoca la compilación del kernel y de sus módulos. También invoca el comando **depmod**, que genera el archivo **modules.dep** de dependencia de módulos.

d. Instalación de módulos

Los módulos se instalan con el comando específico **make modules_install**. Se copian en el directorio **/lib/modules**, en un directorio correspondiente a la versión del kernel.

Visualización de los directorios que contienen los módulos

Cada versión de kernel instalado tiene su directorio de módulos correspondiente.

```
root@servidor:~$ ls /lib/modules/
2.6.27-11-generic  2.6.28-16-generic  2.6.32-22-generic
2.6.27-7-generic   2.6.31-21-generic  2.6.32-23-generic
2.6.27-9-generic   2.6.32-21-generic  2.6.32-24-generic
root@servidor:~$
```

e. Instalación del kernel

El kernel sin módulos se encuentra en el directorio de fuentes en la ruta relativa **arch/x86/boot** para las versiones de 32 bits o **arch/ia64/boot** para las versiones de 64 bits con el nombre **bzImage**. Su instalación en el sistema en producción se realiza copiando simplemente este archivo en el directorio **/boot**. Se puede usar perfectamente el nombre por defecto (bzImage), pero es recomendable renombrarlo para tener en cuenta la versión compilada.

Las compilaciones que se realicen con versiones antiguas de kernel pueden generar un archivo zImage y no un bzImage. El prefijo z o bz indica si el formato de compresión del archivo de kernel es gzip (z) o bzip2 (bz).

Un kernel recién compilado siempre debe instalarse añadiéndose al kernel existente. Nunca reemplace un kernel que funciona por un kernel nuevo.

Instalación del kernel

Las buenas prácticas recomiendan que el archivo de kernel tenga un nombre estándar que refleje su versión.

```
root@servidor# cp arch/x86/boot/bzImage /boot/vmlinuz-2.6.15
root@servidor#
```

f. Creación del ramdisk de módulos

Hay que dejar a disposición del kernel un ramdisk que contenga el conjunto de módulos compilados para la nueva versión. Este ramdisk necesita un archivo de imagen, que puede construirse con dos comandos distintos en función de la generación del sistema usado. El comando tradicional es **mkinitrd**. Este tiende a desaparecer en beneficio del comando **mkinitramfs**.

Creación de un ramdisk con el comando mkinitrd

```
mkinitrd nombre_imagen versión
```

Creación de un ramdisk con el comando mkinitramfs

```
mkinitramfs -o nombre_imagen versión
```

Donde *nombre_imagen* representa el nombre del archivo de imagen de ramdisk que se desea crear y *versión* el número de versión del kernel. Este número se corresponde de hecho con el directorio de módulos albergado en /lib/modules.

Ejemplo de creación de un ramdisk

```
root@servidor:/boot$ mkinitrd /boot/initrd-2.6.28.img 2.6.28
root@servidor:/boot$ file initrd-2.6.28.img
initrd.img-2.6.32-24-generic: gzip compressed data, from Unix, last modified: Fri
Aug 20 07:54:31 2010
root@servidor:/boot$
```

El archivo ramdisk es de hecho un archivo cpio comprimido en formato gzip. Los comandos de creación de ramdisk generan directamente sus archivos en este formato.

Un sistema reciente solo debería ofrecer el comando mkinitramfs; sin embargo, si mkinitrd también está disponible, no debería usarse. mkinitrd se basa en devfs y no en udev, y no soporta los discos sata.

g. Configuración del gestor de arranque

No basta con tener compilado el kernel y haberlo puesto en el sitio adecuado, todavía falta por configurar el gestor de arranque para que sea capaz de cargar este kernel. Por consiguiente, hay que añadir su entrada al gestor de arranque. Atención, no hay que quitar nada de lo que ya haya en la configuración del gestor de arranque: no se toca lo que ya funciona. Basta con añadir una entrada en el archivo de configuración del gestor basándose, si fuera necesario, en las entradas ya existentes.

Añadir una entrada en el gestor de arranque

La conservación de entradas existentes siempre permitirá poder arrancar usando una configuración estable.

```
# kernel operativo original
title           Debian GNU/Linux, kernel 2.6.26-2-686
root            (hd0,0)
kernel          /boot/vmlinuz-2.6.26-2-686 root=/dev/hda1 ro quiet
initrd          /boot/initrd.img-2.6.26-2-686

# kernel añadido para probar
title           PRUEBA - módulos estáticos
root            (hd0,0)
kernel          /boot/vmlinuz-2.6.20 root=/dev/hda1 ro quiet
initrd          /boot/initrd.img-2.6.20
```